

Expanding the Action Space of LLMs to Reason Beyond Language

Zhongqi Yue^{1*} Weishi Wang^{2*} Yundaichuan Zhan³
Juncheng Li³ Daniel Dahlmeier² Fredrik D. Johansson¹

¹Chalmers University of Technology and University of Gothenburg

²SAP ³Zhejiang University

Abstract

Large Language Models (LLMs) are powerful reasoners in natural language, but their actions are typically confined to outputting vocabulary tokens. As a result, interactions with external environments—such as symbolic operators or simulators—must be expressed through text in predefined formats, parsed, and routed to external interfaces. This overloads the model’s language with both reasoning and control duties, and requires a hand-crafted parser, external to the LLM. To address this, we decouple environment interactions from language by internalizing them in an Expanded Action space (ExpA), beyond the vocabulary. The model starts reasoning in the default language environment, but may trigger routing actions and switch to an external environment at any time. From there, the model can only invoke environment-specific actions, receive feedback from the environment, and potentially route back to language as a result. To promote effective exploration of the expanded action space and new environments, we introduce ExpA Reinforcement Learning (EARL) with counterfactual policy optimization. On tasks requiring multi-turn interactions and contingent planning, EARL outperforms strong baselines with vocabulary-constrained actions. It performs robustly across calculator-based multi-task learning and, in the partially observed sorting problem, achieves perfect Sort-4 accuracy while self-discovering an efficient algorithm competitive with classical designs.

1 Introduction

Recent progress has transformed Large Language Models (LLMs) from pure language reasoners into agents that interact with external environments such as tools, APIs, and embodied systems [39, 9, 36]. External environments both augment LLMs with missing capabilities—such as symbolic computation [26] or up-to-date knowledge [38]—and extend their reasoning by mapping instructions into concrete operations like API calls or robotic control [36, 27, 43, 49].

Existing works treat LLMs as agents whose actions are limited to outputting vocabulary tokens $\mathcal{V}$ (Figure 1a). Interactions with external environments rely on parsers that map predefined text patterns (*e.g.*, tool tags or JSON) into environment-specific actions [38], which return plain-text observations appended to the sequence. Models are

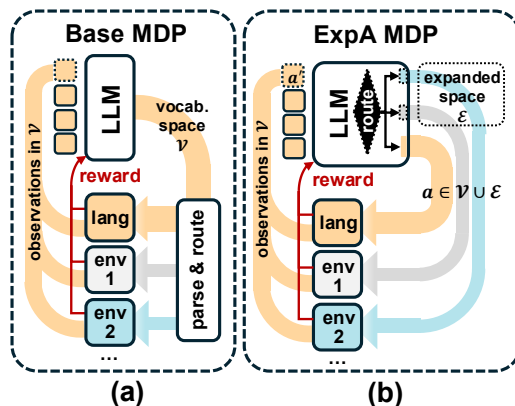


Figure 1: MDP of LLM-environment interaction. (a) Existing works confine actions to vocabulary $\mathcal{V}$, relying on external parsers for interaction. (b) ExpA introduces an expanded action space $\mathcal{E}$ that internalizes environment-specific actions beyond language.

guided by prompts and further trained via supervised tool-call data [38, 31] or reinforcement learning (RL) with rewards from either language outputs [11, 41] or external environments [36].

We propose a fundamental shift from the language-only paradigm. Our aim is threefold: (1) to decouple environment interactions from language reasoning, (2) to enable end-to-end training by removing reliance on external parsers and keeping interactions under the model’s control, (3) to fully support RL on base models, *i.e.*, Zero-RL [9], without requiring supervised tool-call data or adherence to predefined language patterns.

We introduce an Expanded Action space $\mathcal{E}$ (**ExpA**) that allows models to act beyond vocabulary tokens by directly interacting with external environments. In the default language mode, the model either generates tokens from $\mathcal{V}$ or triggers a routing action $g_i \in \mathcal{E}$ to enter environment i (*e.g.*, a calculator). Inside i , it is restricted to environment-specific actions (*e.g.*, calculator buttons) that produce observations in $\mathcal{V}$, and upon completion (*e.g.*, pressing “=”), control returns to the language environment. As shown in Figure 1b, this cleanly separates reasoning from interaction. Unlike simply enlarging the token space [6, 45], ExpA treats external actions as outputs only, avoiding costly input-side fine-tuning and enabling efficient, modular integration of new environments.

To encourage effective intereactions with external environments, we introduce **EARL**, a reinforcement learning framework built on the Expanded Action space (ExpA), which leverages counterfactual rollouts to promote exploration of crucial but rarely used actions. Experiments show consistent gains on multi-turn tasks, including perfect Sort-4 accuracy with a learned efficient algorithm in a partially observed environment.

2 Approach

Problem setting. We consider Large Language Models (LLMs) that interact sequentially with one or more *external environments*, in addition to the default *language environment*. At each step, a model (agent) acts by selecting either a token from the language environment or an action in an external one. We formalize this as a partially observed Markov decision process (POMDP) with a global state $s_t = (h_t, e_t, z_t)$, where h_t is a history of language tokens from a vocabulary $\mathcal{V}$, $e_t \in \{0, 1, \dots, K\}$ denotes the agent’s active environment ($e_t = 0$ for language), and z_t is the latent state of the external environments. The history h_t is fully observed by the agent, comprising a record of tokens selected in the language environment and token-based descriptions of observations from the external environments. Unlike h_t and e_t , z_t is only *partially observed* through interactions.

The agent is represented by a policy $\pi_\theta(a_t | h_t, e_t)$, with parameters θ , that samples an action a_t depending on the observable state and the active environment. Each external environment $i \neq 0$ exposes a step procedure, with a set of permissible set of actions $\mathcal{E}_i$, denoted as $(o_t, z_{t+1}, exit) \leftarrow \text{step}_i(h_t, z_t, a_t)$, which executes $a_t \in \mathcal{E}_i$, produces an observation described by language tokens $o_t \in \mathcal{V}^{(\cdot)}$, an exit flag, and updates the latent state to z_{t+1} . After acting, the agent updates its history by appending the new observation, $h_{t+1} \leftarrow h_t \oplus o_t$ and is routed back to language environment if *exit* is true. At each step, the active environment produces a reward $r_t = R(e_t, h_t, a_t)$, and the agent’s objective is to maximize the cumulative reward $\bar{r} = \sum_t r_t$.

Interaction with expanded action space (ExpA). In this work, we let agents interact directly with environments by expanding their action spaces beyond the vocabulary $\mathcal{V}$. For each environment $i \neq 0$, we add a transition action g_i for entering it and an environment-specific action set $\mathcal{E}_i$ that interfaces with step_i . With $\mathcal{E} = \bigcup_{i=1}^K g_i \cup \mathcal{E}_i$ the set of added actions and $\mathcal{A} = \mathcal{V} \cup \mathcal{E}$ the full action set, the agent’s policy π_θ is a distribution over $\mathcal{A}$, conditioned on the history h_t and active

Algorithm 1 Rollout with ExpA

```

1: Input: policy  $\pi_\theta$ , horizon  $T$ , initial  $s_0 = (h_0, e_0=0, z_0)$ 
2: for  $t = 0, 1, \dots, T - 1$  do
3:   if  $e_t = 0$  then ▷ language environment
4:     Sample  $a_t \sim \pi_\theta(\cdot | h_t, e_t=0)$ 
5:     if  $a_t \in \mathcal{V}$  then
6:        $h_{t+1} \leftarrow h_t \oplus a_t$ ;
7:     else if  $a_t = g_i$  then ▷ route to env  $i$ 
8:        $h_{t+1} \leftarrow h_t \oplus \text{desc}(g_i)$ ;
9:        $e_{t+1} \leftarrow i$ ;
10:    end if
11:  else ▷ external environments
12:    Sample  $a_t \sim \pi_\theta(\cdot | h_t, e_t \neq 0)$ 
13:     $(o_t, z_{t+1}, exit) \leftarrow \text{step}_{e_t}(h_t, z_t, a_t)$ 
14:     $h_{t+1} \leftarrow h_t \oplus \text{desc}(a_t) \oplus o_t$ 
15:    if  $exit=\text{true}$  then
16:       $e_{t+1} \leftarrow 0$  ▷ route back to lang
17:    end if
18:  end if
19:   $r_t \leftarrow R(e_t, h_t, a_t)$ 
20: end for
21: return trajectory  $\tau = \{(h_t, a_t, r_t)\}_{t=0}^{T-1}$ 

```

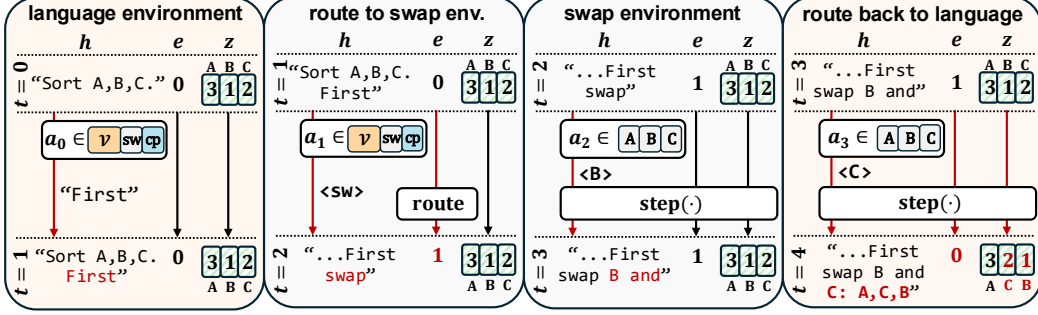


Figure 2: An example rollout with ExpA. Here, $\langle \text{sw} \rangle$ and $\langle \text{cp} \rangle$ route to the swap and compare environments, respectively. Inside them, agents can choose $\langle \text{A} \rangle$, $\langle \text{B} \rangle$, $\langle \text{C} \rangle$ as operands. After two operands are chosen, the step procedure updates the latent state z when necessary, routes back to the language environment, and returns the swap or comparison result as a plain-text observation.

environment e_t . Algorithm 1 describes interactions under this paradigm. Every rollout begins in the language environment ($e_0 = 0$), where the policy π_θ may either select a vocabulary token $a_t \in \mathcal{V}$ to extend the history, or a transition action g_i that routes control to environment i while appending a description of g_i to h_t , denoted as $\text{desc}(g_i)$. Once inside environment i , the policy chooses actions from $\mathcal{E}_i$, triggering the corresponding step $_i$, which outputs an observation described in $\mathcal{V}$, updates the latent state, and returns an additional *exit* flag. If *exit* = true, the environment resets to $e_{t+1} = 0$, routing control back to the language environment. We illustrate with an example in Figure 2.

Training. We train ExpA with Reinforcement Learning (EARL), and propose Counterfactual Policy Optimization (CPO), which optimizes the following objective:

$$\mathcal{J}_{\text{CPO}}(\theta) = \mathbb{E}_{s_0, \mathcal{T}(s_0)=\{(\tau_i, \tau'_i)\}_{i=1}^m} \left[\frac{1}{m} \sum_{i=1}^m U_i(\mathcal{T}(s_0); \theta) \right], \quad (1)$$

where s_0 is an initial state sampled from the training data, $\mathcal{T}(s_0)$ comprises m rollout pairs, and $U_i(\cdot; \theta)$ denotes the update function for the i -th pair. Each τ_i is a *factual* rollout obtained by inputting s_0 into Algorithm 1, while each τ'_i is a *counterfactual* rollout obtained by forcing a routing action at a plausible intermediate step in τ_i . We describe our parameterization of the policy π_θ and the design of the update function in Appendix.

One key challenge is that the policy may fail to reliably invoke routing actions when needed. For instance, a pretrained model has no prior experience of invoking a calculator and thus may not assign high probability to its routing action (e.g., $g_{\text{calc}} = \langle \text{calculate} \rangle$), even when complex arithmetic is required. We address this with *counterfactual rollouts*, which evaluate *what would have happened* had the policy taken a routing action at a plausible intermediate step, thereby encouraging exploration of rarely invoked but critical decisions.

Given a factual rollout $\tau = \{(h_t, a_t, r_t)\}_{t=0}^{T-1}$, we construct a counterfactual rollout τ' as follows: 1) Select a routing action $g_i \in \mathcal{E}$ to be encouraged (e.g., g_{calc} for arithmetic tasks); 2) Sample a time step $t' \in \{t \mid e_t = 0\}$ with weight proportional to $\pi_\theta(\text{desc}(g_i) \mid h_t, e_t = 0)$; 3) Initialize $\tau'_t \leftarrow \tau_t$ using the factual rollout for $t = 1, \dots, t'$; 4) Intervene with $\text{do}(a_{t'} = g_i)$ at t' and apply the transition in Algorithm 1; 5) Continue rollout for $t = t' + 1, \dots, T - 1$ with Algorithm 1 to obtain τ' .

This relies *only* on the pretrained next-token distribution (step 2). For example, if $\text{desc}(g_{\text{calc}}) = \langle \text{calculate} \rangle$, the insertion “To solve it, first calculate” is more probable under the language model than “To solve calculate, ...” and is weighed more heavily when forcing a routing action. Hence the method is fully compatible with zero-RL training [9, 52].

3 Experiment

Setup. We evaluate tasks that require multi-turn interactions with external environments. Prior work has focused mainly on math problems [11] or API calls [36], where the solution path follows a *fixed* derivation. We additionally target the harder problem of *contingent planning*, where the agent must adapt its strategy based on intermediate observations. We design two complementary tasks:

Example Questions		Environments
Arithmetic	What is subtraction of 69 divided by 6 plus 7049 plus 13 plus 50 plus 643 from 9936?	Calculator +-×÷= ()1234567890.
Countdown	Using the numbers [2900, 1205, 5911, 4], create an equation that equals -4212.	Compare A, B, C, D, E
GSM8K*	Henry made two stops in a 691206-mile trip...How many miles did he go in the first stop?	Swap A, B, C, D, E
Count	How many times does the digit 0 appear in the number fifty-eight billion, thirty million...?	
Sorting	Sort the following items in ascending order: A, B, C, D, E.	

Figure 3: Example questions and the environment-specific actions in **Calc-Bench** and **Sorting**.

Table 1: Results (EM) on Calc-Bench, we jointly train our models on all Calc-Bench tasks to assess the benefits of shared representation learning.

Method	Calc Bench				
	Arithmetic	Countdown	Count	GSM8K*	Overall
GPT-4o	41.30	18.85	66.85	31.95	39.74
Qwen-2.5-3B-Instruct	15.80	2.80	66.50	20.55	26.41
SFT+GRPO	70.75	48.50	93.85	30.57	60.92
Prompt+GRPO	64.70	49.15	94.75	30.39	59.75
Prompt+CPO	61.50	38.30	91.35	46.80	59.49
ExpA+CPO (EARL)	69.20	75.15	93.70	48.53	71.65
Qwen-2.5-7B-Instruct	22.60	11.75	74.05	24.01	33.10
SFT+GRPO	56.00	66.70	93.00	34.20	62.48
Prompt+GRPO	80.30	60.70	98.60	33.33	68.23
Prompt+CPO	64.85	55.15	94.55	52.33	66.72
ExpA+CPO (EARL)	78.10	84.25	98.70	53.71	78.69

- **Calc-Bench.** The agent uses a stateless *calculator* environment (Figure 3), with rewards based on exact-match (EM) accuracy. The benchmark spans four datasets (statistics in Appendix, examples in Figure 3): (1) *Arithmetic*, which tests calculator use under out-of-distribution settings where operations or numbers may appear in natural language; (2) *Countdown*, which stresses contingent planning by requiring efficient adaptation across up to 7,680 possible combinations; (3) *GSM8K**, an upscaled variant of GSM8K [8] with larger numbers but preserved semantics; and (4) *Count*, which checks whether agents retain basic numerical understanding while learning (1)–(3).
- **Sorting.** The agent must order a set of *hidden* numbers using *compare* and *swap* environments; e.g., “compare A, B” reveals their relation, while “swap A, B” updates the hidden state. Rewards depend on correctly sorting the hidden state z_T , with penalties for excessive operations. This task is challenging as it forms a POMDP: the agent must plan contingently from comparison outcomes, reason precisely over first-order relations, and manipulate hidden states through interactions rather than text output. Training uses Sort-2 to Sort-5, while evaluation is on Sort-4 and Sort-5.

Baselines. We compare against four paradigms: (1) *SFT+GRPO*, fine-tuning on labeled interactions followed by GRPO [39, 11]; (2) *Prompt-GRPO* [9, 41], which provides interaction patterns in the prompt and trains with GRPO; (3) *Prompt-CPO*, identical to Prompt-GRPO but using our Counterfactual Policy Optimization (CPO); and (4) *Zero-shot*, proprietary models like GPT-4o [18].

Calc-bench results. As shown in Table 1, zero-shot models perform poorly on this challenging benchmark, with GPT-4o reaching only 39.74 overall EM. Training with a calculator greatly improves performance, but baselines remain inconsistent across tasks and perform especially poorly on contingent-planning tasks like Countdown. In contrast, EARL delivers strong results across all tasks, with up to 10.73% absolute gain overall and as much as 26% gain on Countdown. We analyze in Appendix how the proposed ExpA and CPO help achieve these large gains.

Sorting results. Figure 4 reports sorting accuracy, stratified by the minimum number of swaps required. On Sort-4, EARL achieves perfect accuracy across all levels, whereas Prompt+CPO degrades as the required number of swaps increases. The gap widens on the more challenging Sort-5 problems, where EARL outperforms the best baseline (Prompt+GRPO) by up to 21% in a stratum and more than 10% overall.

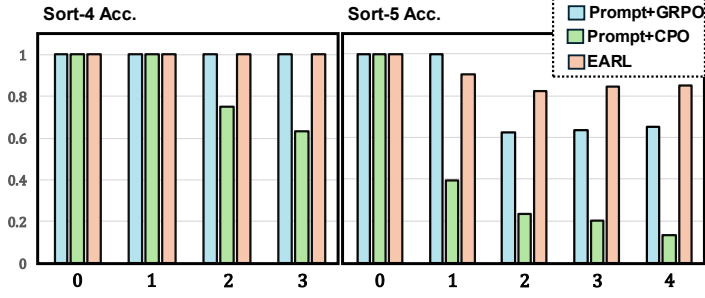


Figure 4: Sort accuracy stratified by the number of required swaps.

Table 2: Average number of swaps (SW) and comparisons (CP) to sort 4 random numbers.

Method	#SW	#CP
Prompt+GRPO	2.076	6.101
EARL	1.917	5.708
EARL*	1.917	4.833
GCC	2.333	5.000
Insert-sort	3.000	4.917
Optimal	1.917	4.667

Efficiency. We assess efficiency by measuring the average comparisons and swaps needed to sort random numbers. With greedy decoding, EARL executes a deterministic decision tree (visualized in the appendix). Pruning redundant steps yields a simplified variant, EARL* (Algorithm 2). The algorithm pivots on A , performs only necessary checks, and applies $\text{MIN_SWAP}(\mathcal{R})$ for minimal swaps. As shown in Table 2, both EARL and EARL* match the optimal number of swaps and nearly achieve the optimal comparisons, surpassing insertion sort and even GCC’s built-in routine.

Sorting with RL. Our study connects to recent work on discovering faster sorting algorithms with RL, most notably AlphaDev [32]. A key distinction is that we leverage the LLM’s natural language vocabulary to represent context and chain reasoning steps, rather than relying on dedicated symbolic states or low-level assembly instructions. This means that our agent is more general-purpose, capable of reusing pre-trained language knowledge. Consequently, EARL achieves 100% accuracy on Sort-4 after only ~ 70 training steps, compared to the million-step training required by AlphaDev, underscoring the value of transferring language reasoning into interactive environments. While performance on Sort-5 is not yet perfect, our goal is to demonstrate how ExpA improves reasoning with external environments, leaving dedicated algorithm discovery and more challenging settings (e.g., VARSORT) as promising future work.

Algorithm 2 EARL* Sort-4

```

1: Input: four numbers  $A, B, C, D$ 
2:  $\mathcal{R} \leftarrow \emptyset$ 
3:  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(A, B)\}$ 
4:  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(A, C)\}$ 
5:  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(A, D)\}$ 
6: if not  $(C < A < B \vee B < A < C)$  then
7:    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(B, C)\}$ 
8: end if
9: if not  $(D < A < B \vee B < A < D)$  then
10:   $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(B, D)\}$ 
11: end if
12: if not  $(D < A < C \vee C < A < D)$  then
13:   $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{Compare}(C, D)\}$ 
14: end if
15:  $\text{MIN\_SWAP}(\mathcal{R})$ 

```

4 Conclusion

We presented **ExpA**, a paradigm that equips LLMs with expanded action space for environment interactions. ExpA decouples reasoning from interaction, eliminating reliance on external parsers and enabling end-to-end training. To optimize in this setting, we proposed **EARL**, a counterfactual RL method that encourages exploration of rarely used but crucial actions. Experiments show consistent gains over strong baselines, especially on contingent planning tasks, and even the discovery of an efficient Sort-4 algorithm. Overall, ExpA and EARL provide a scalable framework for reasoning beyond language, opening paths toward mathematical reasoning and large-scale zero-RL tool use.

5 Acknowledgement

Zhongqi Yue is supported by the Wallenberg-NTU Presidential Postdoctoral Fellowship. Fredrik D. Johansson is supported in part by the Wallenberg AI, Autonomous Systems and Software Program funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

- [1] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nat. Mac. Intell.*, 6(5):525–535, 2024.
- [2] Yash Chandak, Georgios Theodorou, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pages 941–950. PMLR, 2019.
- [3] Yash Chandak, Georgios Theodorou, Chris Nota, and Philip S. Thomas. Lifelong learning with a changing action set. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3373–3380. AAAI Press, 2020.
- [4] Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. Learning evolving tools for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- [5] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2023.
- [6] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.
- [7] Yun-Shiuan Chuang, Agam Goyal, Nikunj Harlalka, Siddharth Suresh, Robert Hawkins, Sijia Yang, Dhavan Shah, Junjie Hu, and Timothy T. Rogers. Simulating opinion dynamics with networks of llm-based agents. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3326–3346. Association for Computational Linguistics, 2024.
- [8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.
- [9] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.
- [10] Gregory Farquhar, Laura Gustafson, Zeming Lin, Shimon Whiteson, Nicolas Usunier, and Gabriel Synnaeve. Growing action spaces. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3040–3051. PMLR, 2020.

- [11] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.
- [12] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR, 2023.
- [13] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [14] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [15] Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. Middleware for llms: Tools are instrumental for language agents in complex environments. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7646–7663. Association for Computational Linguistics, 2024.
- [16] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [17] Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. Solving math word problems by combining language models with symbolic solvers. *CoRR*, abs/2304.09102, 2023.
- [18] Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispositi, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pélisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Gierler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll L. Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, and Dane Sherburn. Gpt-4o system card. *CoRR*, abs/2410.21276, 2024.
- [19] Ayush Jain, Norio Kosaka, Kyung-Min Kim, and Joseph J Lim. Know your action set: Learning action relations for reinforcement learning. In *International Conference on Learning Representations*, 2021.
- [20] Ayush Jain, Norio Kosaka, Kyung-Min Kim, and Joseph J. Lim. Know your action set: Learning action relations for reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

- [21] Ayush Jain, Andrew Szot, and Joseph J. Lim. Generalization to new actions in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4661–4672. PMLR, 2020.
- [22] Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. Genegpt: augmenting large language models with domain tools for improved access to biomedical information. *Bioinform.*, 40(2), 2024.
- [23] Yizhao Jin, Greg G. Slabaugh, and Simon M. Lucas. ADAPTER-RL: adaptation of any agent using reinforcement learning. *CoRR*, abs/2311.11537, 2023.
- [24] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tennenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *CoRR*, abs/2205.00445, 2022.
- [25] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.
- [26] Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. In *International Conference on Learning Representations*. ICLR 2024, 2024.
- [27] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, 2023.
- [28] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 9493–9500. IEEE, 2023.
- [29] Zhaoyang Liu, Yinan He, Wenhai Wang, Weiyun Wang, Yi Wang, Shoufa Chen, Qing-Long Zhang, Yang Yang, Qingyun Li, Jiashuo Yu, Kunchang Li, Zhe Chen, Xuecheng Yang, Xizhou Zhu, Yali Wang, Limin Wang, Ping Luo, Jifeng Dai, and Yu Qiao. Interngpt: Solving vision-centric tasks by interacting with chatbots beyond language. *ArXiv*, abs/2305.05662, 2023.
- [30] Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *CoRR*, abs/2502.11271, 2025.
- [31] Ne Luo, Aryo Pradipta Gema, Xuanli He, Emile Van Krieken, Pietro Lesci, and Pasquale Minervini. Self-training large language models for tool-use without demonstrations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1253–1271, 2025.
- [32] Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.
- [33] Amogh Mannekote, Adam Davies, Jina Kang, and Kristy Elizabeth Boyer. Can llms reliably simulate human learner actions? A simulation authoring framework for open-ended learning environments. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 29044–29052. AAAI Press, 2025.
- [34] Aaron Parisi, Yao Zhao, and Noah Fiedel. TALM: tool augmented language models. *CoRR*, abs/2205.12255, 2022.

- [35] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi R. Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Guoliang Li, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models. *ACM Comput. Surv.*, 57(4):101:1–101:40, 2025.
- [36] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [37] James Queeney, Xiaoyi Cai, Mouhacine Benosman, and Jonathan P. How. GRAM: generalization in deep RL with a robust adaptation module. *CoRR*, abs/2412.04323, 2024.
- [38] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [39] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.
- [40] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [41] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.
- [42] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 11854–11864. IEEE, 2023.
- [43] Andrew Szot, Bogdan Mazouze, Omar Attia, Aleksei Timofeev, Harsh Agrawal, R. Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 10644–10655. Computer Vision Foundation / IEEE, 2025.
- [44] Adrian Theuma and Ehsan Shareghi. Equipping language models with tool use capability for tabular data analysis in finance. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - Volume 2: Short Papers, St. Julian’s, Malta, March 17-22, 2024*, pages 90–103. Association for Computational Linguistics, 2024.
- [45] Bohan Wang, Zhongqi Yue, Fengda Zhang, Shuo Chen, Li’an Bi, Junzhe Zhang, Xue Song, Kennard Yanting Chan, Jiachun Pan, Weijia Wu, et al. Selftok: Discrete visual tokens of autoregression, by diffusion, and for reasoning. *arXiv preprint arXiv:2505.07538*, 2025.
- [46] 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. *Trans. Mach. Learn. Res.*, 2024, 2024.
- [47] Jingwei Wang, Zai Zhang, Hao Qian, Chunjing Gan, Binbin Hu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Bin Shi, and Bo Dong. Enhancing LLM tool use with high-quality instruction data from knowledge graph. *CoRR*, abs/2506.21071, 2025.

- [48] Qian Wang, Jiaying Wu, Zhenheng Tang, Bingqiao Luo, Nuo Chen, Wei Chen, and Bingsheng He. What limits llm-based human simulation: Llms or our design? *CoRR*, abs/2501.08579, 2025.
- [49] Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36:75392–75412, 2023.
- [50] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.
- [51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [52] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.
- [53] Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. Evaluating and improving tool-augmented computation-intensive math reasoning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [54] Wanpeng Zhang and Zongqing Lu. Rladapter: Bridging large language models to reinforcement learning in open worlds. *CoRR*, abs/2309.17176, 2023.
- [55] Haiteng Zhao, Chang Ma, Guoyin Wang, Jing Su, Lingpeng Kong, Jingjing Xu, Zhi-Hong Deng, and Hongxia Yang. Empowering large language model agents through action learning. *CoRR*, abs/2402.15809, 2024.

A Appendix

B Related Work

Recent advances demonstrate LLMs as powerful reasoners in natural language [51, 40, 38]. Many works have extended their role to agents interacting with external environments, and consider tasks such as tool utilization [34, 30, 35], multi-modality interpretability [55, 42, 46], math reasoning [24, 17, 53], program-guided reasoning [14, 12, 5, 28], real-time knowledge integration [47, 13, 15], and domain-specific scenarios [1, 22, 44]. However, most existing approaches require the model to express task-specific actions as predefined text patterns, which are then parsed and routed to external environments. This design relies heavily on the model’s instruction-following ability [16], making performance highly sensitive to prompt variations [33] and dependent on pre-trained knowledge for action execution [16]. Moreover, many methods require human-crafted demonstrations of tool usage [4, 29], further limiting scalability. In contrast, EARL endows agents with new capabilities by introducing environment-specific actions, explored and learned through counterfactual policy optimization without human demonstration.

While prior works have not explored expanding the action space of LLMs, expanding the *token space* is common in multimodal LLMs, which typically requires large-scale training on multimodal demonstrations, sometimes combined with online RL [43]. Another related direction introduces action *adaptors*, which constrain the model to a set of learned actions tailored for a specific environment [7, 48]. This can be viewed as a simplified version of Figure 1b, involving only one external environment and no language environment. Beyond LLMs, growing action spaces have been studied to accelerate exploration [10] or to extend the set of available actions in a single environment [21, 20, 3]. Continual RL research [25, 23, 54] has also demonstrated the effectiveness of action learning in non-stationary settings [2, 37]. In contrast, our work considers the more general and challenging case where LLMs must reason in language while sequentially interacting with multiple external environments. To this end, we expand the action space of LLMs and propose a reinforcement learning algorithm for efficient exploration of external interactions.

C Additional Details of Setting

Language-only interaction. In existing approaches to tool use and external interaction [11, 41], agents never truly leave the language environment: they always select actions $a_t \in \mathcal{V}$ from their own vocabulary, and extend the observation history as $h_{t+1} = h_t \oplus a_t$. Interacting with an external environment $i \neq 0$ requires *translating* h_{t+1} to actions in $\mathcal{E}_i$. Typically, this is realized by detecting predefined patterns, such as `<calculator>...</calculator>` or structured JSON fields. When a pattern that indicates interaction with environment i appears in h_{t+1} , its contents are *parsed* into a sequence of actions from $\mathcal{E}_i$ that get executed by step i . A drawback is that no intermediate feedback can affect the choice of actions within the pattern (e.g., `<calculator>` block), which may hamper performance and credit assignment.

D EARL

D.1 A Policy over the Expanded Action Space

A central challenge in operating with expanded action spaces is how to represent and generalize to the newly introduced actions. Prior work points to two guiding principles: First, the policy should condition on the set of available actions [21, 19]. Second, prior knowledge about actions can be leveraged to improve generalization, through learned action embeddings [19] or by incorporating known structure in training [10]. We adopt both.

Policy parameterization. To condition the policy on all available actions, we extend the standard LLM classification head. In the language-only setting, the head produces $|\mathcal{V}|$ logits over the vocabulary. With ExpA, this head is expanded to output $|\mathcal{V} \cup \mathcal{E}|$ logits. We denote by θ the parameters of the LLM together with the expanded head. At step t , the encoded feature of h_t is projected to logits, and a softmax is applied over the subset of actions available in environment e_t , yielding $\pi_\theta(\cdot \mid h_t, e_t)$.

Policy initialization. Each action $a \in \mathcal{E}$ has a natural language description $\text{desc}(a)$, such as the environment name (e.g., “calculator”) or the semantic label of a step procedure (e.g., “compare”). To

exploit this prior knowledge about action similarities, we initialize the weights of new actions so that selecting an action has approximately the same likelihood as producing its description:

$$\pi_{\theta}(a \mid h_t, e_t) \approx \pi_{\theta}(\text{desc}(a) \mid h_t, e_t),$$

where e_t is the active environment at t . In particular, when the description is a single token, this condition can be satisfied directly (and exactly) by initializing the new action weight with the pretrained weight of the token $\text{desc}(a)$. This aligns expanded actions with their linguistic counterparts from the start, providing a strong prior that accelerates learning.

D.2 Update Function

Finally, we define the update function for each rollout pair (τ_i, τ'_i) , $i \in \{1, \dots, m\}$, as

$$U_i(\mathcal{T}(s_0); \theta) = \begin{cases} f(\tau'_i, \bar{r}'_i - \bar{r}_i; \theta), & \text{if } \bar{r}_j \leq 0 \ \forall j \in \{1, \dots, m\}, \\ f(\tau_i, \frac{\bar{r}_i - \mu}{\sigma}; \theta), & \text{otherwise,} \end{cases}$$

where $\bar{r}_i$ and $\bar{r}'_i$ denote the cumulative rewards of τ_i and τ'_i , respectively, and μ, σ are the mean and standard deviation of rewards across the factual rollouts. Here $f(\tau, a; \theta)$ denotes the standard update rule [39], which takes as input a rollout trajectory τ and its associated advantage scalar, and applies PPO-style clipping and KL regularization (details in Appendix). The design is motivated by balancing exploration and exploitation: when the current rollout fails to achieve positive reward, the first counterfactual branch encourages exploration of missing interactions; otherwise, the update reduces to group-relative advantage, exploiting successful strategies.

D.3 Update Rule

The update rule is given by

$$f(\tau, a; \theta) = \sum_{t=0}^{T-1} \left[\min(r_t(\theta) a, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) a) - \beta \text{KL}(\pi_{\theta}(\cdot \mid h_t) \parallel \pi_{\text{ref}}(\cdot \mid h_t)) \right], \quad (2)$$

where

- $\tau = \{(h_t, a_t, r_t)\}_{t=0}^{T-1}$ is a rollout trajectory,
- h_t denotes the token history (state) at step t ,
- $r_t(\theta) = \frac{\pi_{\theta}(a_t \mid h_t)}{\pi_{\text{old}}(a_t \mid h_t)}$ is the importance sampling ratio between the new and old policies,
- ϵ is the PPO clipping threshold,
- β is the KL regularization coefficient,
- $\pi_{\theta}(\cdot \mid h_t)$ and $\pi_{\text{ref}}(\cdot \mid h_t)$ denote the current and reference policies, respectively. We use the pre-trained LLM as the reference model.

Note that we slightly abuse notation in the main paper by saying τ is generated with π_{θ} . In practice, rollouts are generated by the reference policy π_{old} , which is held fixed during data collection. The objective in (2) then compares the likelihood of these sampled actions under the current policy π_{θ} versus the reference policy π_{old} , with the ratio $r_t(\theta)$ providing the necessary importance weighting. This distinction ensures stable on-policy learning: trajectories are collected with π_{old} , while updates adjust π_{θ} to maximize advantage without diverging too far from π_{old} .

In training, we do not perform update on positions in each rollout corresponding to observations returned by external environments. For EARL, we apply the KL loss with token probabilities computed over the original vocabulary space $\mathcal{V}$.

E Additional Dataset Details

Calc-Bench Dataset Details. The *Calc-Bench* benchmark consists of four sub-datasets targeting different types of mathematical reasoning: *Arithmetic*, *Countdown*, *Count*, and a rewritten subset of GSM8K, denoted *GSM8K**. Each dataset is generated via task-specific scripts, with explicit control over complexity, number format, and structure.

Table 3: Statistics of the Calc-Bench datasets. Language portions refer to the portion of questions where operations or numbers are written in natural language.

Task	Max number (10^x)		#Operands		Lang. portion		#Instances	
	Train	Test	Train	Test	Train	Test	Train	Test
Arithmetic	5	5	5	7	10%	70%	1,000	2,000
Countdown	4	4	4	4	NA	NA	20,000	2,000
GSM8K*	6	6	NA	NA	NA	NA	5,317	579
Count	20	20	NA	NA	90%	90%	1,000	2,000

- *Arithmetic*: This dataset contains randomly generated arithmetic expressions involving up to 5-digit integers and between 2 and 6 operations. The training set includes 1,000 samples with a maximum of 4 operations and 10% of examples paraphrased into natural language. The test set contains 2,000 samples with up to 6 operations and 70% paraphrased into natural language.
- *Countdown*: Each instance is generated by sampling a valid arithmetic expression, extracting all numerical values, shuffling them, and using the original result as the target. The final dataset includes 20,000 training and 2,000 test examples. Expressions contain up to 4-digit numbers and allow up to 3 operations. Each number must appear exactly once in the constructed expression.
- *GSM8K**: This subset is derived from a manually rewritten version of GSM8K, filtered to include only examples explicitly marked as rewritten. Each sample includes a paraphrased question and corresponding reasoning-based answer. This version preserves the complexity of GSM8K while reducing lexical overlap with the original dataset.
- *Count*: This dataset consists of symbol sequences with a maximum length of 20. Each example is labeled with a count-based target (e.g., the number of specific items). The training set includes 1,000 examples and the test set includes 2,000.

Sorting Experimental Setup. We adopt a curriculum-based training strategy for sorting tasks. Specifically, we first use an *ordering* dataset to pre-train the model on simpler relational tasks, followed by a *sorting* dataset that introduces the full sorting items. Each dataset is constructed with its own generation procedure and input length distribution, as detailed below.

- *Ordering*: A total of 20,000 training examples are generated, with 95% assigned to the *order* task and 5% to *compare*. Each input sequence contains 2 to 5 items, with a length distribution of 30% for 2 items, 30% for 3 items, 20% for 4 items, and 20% for 5 items. The corresponding test set contains 2,000 *order* examples, with the same item length distribution.
- *Sorting*: The training set contains 80,000 examples, with input lengths ranging from 2 to 5 elements, denoted as Sort-2 through Sort-5. The distribution is 10% Sort-2, 20% Sort-3, 30% Sort-4, and 40% Sort-5. The test set includes 2,000 examples, equally split between Sort-4 and Sort-5 cases.

Implementation Details. We use the open-source Qwen2.5 [50] as our backbone, including results for both base models and instruction-tuned variants with 0.5B, 3B, and 7B parameters. The maximum sequence length is set to 1,024 for Calc-Bench and 384 for Sorting. Training is performed on NVIDIA A100-80GB GPUs, with 1, 2, and 4 GPUs allocated for the 0.5B, 3B, and 7B models, respectively. For fair comparison, we follow standard hyperparameters and optimization protocols [41] (e.g., KL regularization weight, PPO clipping threshold) for both baselines and our method.

F Additional Results

Countdown analysis. We provide a detailed analysis of the results on Countdown with the 3B model in Table 4, supplemented by case studies in the appendix. Several observations emerge:

- Prompt+GRPO triggers the most calculator interactions and avoids hallucinations. However, it often degenerates into inefficient brute-force trials, showing limited use of planning cues after observations (e.g., “this is far from target”).
- Replacing GRPO with CPO increases the use of planning keywords, likely because counterfactual interventions provide more training signals on how to react to observations. Yet this also introduces

Table 4: Average occurrence of external interactions, hallucinations and planning phrases in a validation rollout on Countdown task. We define hallucination as inputting a number that is not given in the question to the calculator. We identify a list of planning phrases commonly used by models in the appendix.

Method	#interactions	#hallucinations	#plannings
EARL	11.25	0.001	20.48
Prompt-CPO	4.85	0.7424	3.22
Prompt-GRPO	15.10	0.000	1.84
SFT-GRPO	3.16	0.6885	1.96

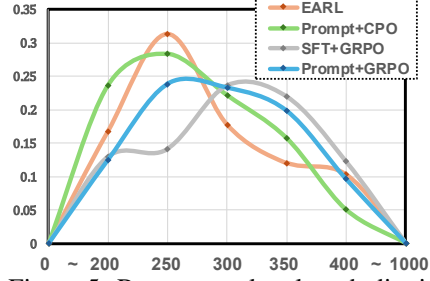


Figure 5: Response token length distribution for correct rollouts in GSM8K*.

Table 5: Ablation on Countdown: CPO vs. GRPO, training on Qwen-Instruct vs. -Base, and with (w/) vs. without (w/o) environment prompt (env.p). ExpA+CPO corresponds to EARL.

	Instruct		Base	
	w/ env.p	w/o	w/ env.p	w/o
ExpA+CPO	80.09	76.76	77.31	74.56
ExpA+GRPO	75.10	73.79	76.45	70.27
Prompt+CPO	67.23	-	63.64	-
Prompt+GRPO	58.16	-	51.15	-
SFT+GRPO	62.05	-	61.17	-

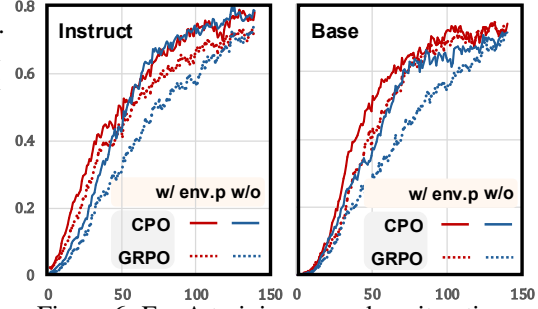


Figure 6: ExpA training reward vs. iteration.

hallucinations, where the agent invents numbers not in the problem. A plausible cause is interference between language reasoning and action learning, *e.g.*, the KL penalty helps preserve pre-trained language knowledge but may hinder the grounding of calculator actions.

- SFT+GRPO uses the fewest interactions, sometimes performing parts of a computation in language mode and then feeding incorrect results to the calculator, causing hallucinations. This suggests that SFT-learned patterns transfer poorly to diverse problem instances.
- EARL uses a moderate number of interactions but produces by far the most planning-related language, yielding much stronger results than all baselines in Table 1. This strongly validates the benefit of decoupling environment interactions from language reasoning, which removes confusion between reasoning and action learning and enables effective use of external environments.

GSM8K* analysis. In Figure 5, we show the distribution of response lengths among correct rollouts on GSM8K* with the 3B model. The results reveal a clear link between efficiency in reasoning (fewer tokens) and stronger performance. Notably, the two methods using CPO (EARL and Prompt+CPO) outperform those with GRPO (SFT+GRPO and Prompt+GRPO), underscoring the importance of encouraging diverse environment interactions.

Ablation Study. We perform ablation on the challenging Countdown task from 3 perspectives:

- *CPO vs. GRPO*: As shown in Table 5, CPO consistently outperforms GRPO across all settings, even for baselines (Prompt+CPO). It also converges faster, as seen by comparing each solid line (CPO) with the dotted line of the same color (GRPO) in Figure 6, highlighting the role of counterfactual rollouts in promoting exploration of new environments and their actions.
- *Instruct vs. Base*: With ExpA, even base models achieve competitive performance, whereas baselines algorithms such as Prompt+GRPO degrade sharply without instruction tuning. This suggests strong potential for using EARL in Zero-RL training of agents for interactive problem solving.
- *Prompted vs. unprompted environments*: Prompting the agent on how to interact is essential for prompt+RL baselines. With ExpA, however, models succeed without such prompts by leveraging weight initialization (Section D.1, ExpA+GRPO w/o) and counterfactual rollouts (ExpA+CPO w/o), indicating scalability to settings with large number of environments.