

Learning to Act Beyond Language: Expanded Action Spaces for Sequential Decision-Making with LLM

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

Project page: <https://expa-rl.github.io/>

*Equal contribution. zhongqi@chalmers.se, weishi.wang@sap.com

Abstract

Large Language Models (LLMs) excel when a solution can be planned from the prompt, but many sequential decision-making tasks require actions to depend on observations acquired during interaction. We study two central challenges in this regime: learning reusable interaction policies for history-dependent search, and acting under partial observability when task-relevant state is revealed only through interaction. We instantiate these challenges with two diagnostic benchmarks: Calc-Bench for reusable, history-dependent tool interaction, and partially observed sorting for action selection under hidden state. We introduce the Expanded Action space (ExpA), which decouples language reasoning from environment interaction by adding routing and environment-specific actions beyond vocabulary tokens. ExpA is trained with EARL, an RL algorithm using counterfactual policy optimization to explore rarely invoked but useful actions. EARL improves Countdown by 26.3% with 3B models and achieves perfect Sort-4 accuracy while discovering an efficient sorting strategy, providing a proof-of-concept for learned non-language interaction in controlled sequential decision-making diagnostics.

1 Introduction

Frontier Large Language Models (LLMs) are remarkably capable when a task admits a one-shot solution: write a program, generate a chain-of-thought, or produce a plan that can be verified before execution (Shao et al., 2024; DeepSeek-AI et al., 2025; Chen et al., 2021). Coding benchmarks such as HumanEval and SWE-bench (Jimenez et al., 2024) have seen rapid progress because the required action sequence can often be derived from the task description. In such settings, the agent can plan ahead: the solution is primarily a function of the initial specification.

Many practical agent settings, including web navigation (Zhou et al., 2024), interactive games,

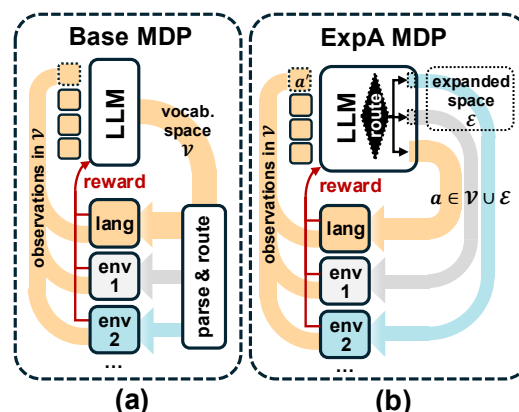


Figure 1: The Markov Decision Process (MDP) of LLM interacting with external environments. (a) In existing works, LLM is confined to its vocabulary space $\mathcal{V}$ for both reasoning and interaction with external environments, where the latter requires an external parser to detect special patterns. (b) We decouple environment interactions from language by internalizing them as an Expanded Action space (ExpA) $\mathcal{E}$ beyond vocabulary.

embodied tasks, and partially observed planning problems, require agents to revise their behavior after observing the environment. This is a central feature of sequential decision-making: agents often need to search over future action paths (Browne et al., 2012) while acting under partial observability, where information-gathering actions shape future decisions (Kaelbling et al., 1998; Sutton and Barto, 2018). Direct evaluation in realistic environments, however, can entangle sequential decision-making with visual perception, web-page understanding, long-context retrieval, or simulator-specific engineering. We therefore use controlled diagnostics to isolate history-dependent interaction: **Calc-Bench**, a calculator-based multitask benchmark that includes **Countdown** as a test of history-dependent search over solution paths; and **partially observed sorting**, a test of action selection under hidden state. These settings are not end applications, but con-

trolled probes of two complementary challenges: search conditioned on intermediate outcomes and decision-making under partial observability.

Existing LLM-agent interfaces typically express environment actions as text, which are then detected and routed by hand-crafted parsers (Schick et al., 2023) (Figure 1a). This makes sequential decision-making a difficult multitask learning problem inside the vocabulary space $\mathcal{V}$: the model must preserve language understanding and planning-ahead abilities while also learning new interaction policies, such as when to query an environment, which action to take, and how to adapt after observing the result. When both reasoning and control are optimized through the same language channel, learning executable interactions can interfere with the very semantic and planning abilities needed to use them effectively.

We therefore separate the space for reasoning from the space for interaction. Our **Expanded Action space (ExpA)** augments vocabulary tokens with routing actions and environment-specific actions. The model can reason in language, route into an external environment when needed, execute environment actions directly, receive observations, and return to language reasoning (Figure 1b). Thus, ExpA turns environment interaction from a parsed text convention into an explicit part of the policy’s action space, allowing history-dependent interaction strategies to be learned separately from vocabulary-only reasoning.

A central challenge remains: pre-trained LLMs have no experience selecting newly introduced actions. We address this with **ExpA Reinforcement Learning (EARL)**, which uses counterfactual policy optimization to encourage exploration of useful but rarely invoked interactions. For each rollout, EARL forces a routing action at a plausible intermediate step and uses the reward difference between factual and counterfactual trajectories as the learning signal. This enables the model to learn when environment interaction improves future outcomes, rather than rediscovering such strategies from prompt instructions in each episode.

In summary, we study controlled sequential decision-making settings where planning ahead is insufficient: agents must learn robust environment interaction and adapt future actions to observed history. Our contributions are:

- **Diagnostic framing:** we isolate complementary forms of history-dependent interaction, includ-

ing reward-path search in Countdown, multitask calculator-mediated reasoning in Calc-Bench, and hidden-state information gathering in partially observed sorting.

- **Expanded Action space (ExpA):** a mechanism that decouples language reasoning from environment interaction by introducing explicit routing and environment-specific actions beyond $\mathcal{V}$.
- **ExpA Reinforcement Learning (EARL):** an RL algorithm with counterfactual rollouts that encourages exploration of rarely invoked but useful environment interactions.
- **Empirical study:** across Calc-Bench and sorting with Qwen-2.5 models, EARL improves multitask calculator use, outperforms vocabulary-constrained baselines on Countdown (*e.g.*, by 26.3%), and achieves perfect Sort-4 accuracy while discovering an efficient sorting strategy.

2 Related Work

LLM tool use and external interaction. Recent work extends LLMs from language reasoners (Yao et al., 2023; Shinn et al., 2023; Schick et al., 2023) to agents that interact with tools, APIs, and environments, including tool use, math and program-guided reasoning, web navigation, knowledge integration, and domain-specific applications (Parisi et al., 2022; Lu et al., 2025; Qin et al., 2025; Karpas et al., 2022; He-Yueya et al., 2023; Zhang et al., 2023; Gou et al., 2024; Gao et al., 2023; Zhou et al., 2024; Wang et al., 2025a; Bran et al., 2024). Most systems express external actions as text patterns (such as tags and JSON fields), or function-call templates, which are parsed and routed to environments. This interface is flexible, but makes interaction sensitive to prompts (Mannekote et al., 2025), dependent on instruction following (Hao et al., 2023), and often reliant on demonstrations (Chen et al., 2025; Liu et al., 2023). In sequential decision-making settings (Song et al., 2024; Zhou et al., 2024; Liu et al., 2024), where useful actions may depend on mid-trajectory observations, ExpA instead represents environment control as direct policy actions beyond vocabulary tokens.

Tool tokens, code generation, and action spaces. Tool-token methods such as ToolkenGPT (Hao et al., 2023) learn tool embeddings and trigger tools by generating special tokens, but remain within the autoregressive token interface and typically rely on supervised tool-use demonstrations. ExpA instead

adds executable actions to the policy head outside the language-token space and trains their use with reinforcement learning. Code-generation methods such as PAL (Gao et al., 2023) and ToRA (Gou et al., 2024) are powerful when the relevant inputs and interaction protocol can be specified before execution; our focus is complementary, targeting settings where future actions depend on observations gathered during the trajectory.

Action spaces beyond language. ExpA also differs from expanding the input token space, as in multimodal LLMs (Szot et al., 2025): external actions are policy outputs, while observations remain ordinary text. Related action-adaptor methods constrain models to learned actions for a specific environment (Chuang et al., 2024; Wang et al., 2025b); ExpA retains the language environment while allowing routing into external environments. Beyond LLMs, growing or changing action spaces have been studied in RL for exploration and continual adaptation (Farquhar et al., 2020; Jain et al., 2020, 2022; Chandak et al., 2020; Khetarpal et al., 2022). Our work brings this perspective to LLM agents and proposes counterfactual policy optimization to explore newly introduced environment actions.

3 Formulation

Problem setting. We consider LLMs acting as agents that interact sequentially with one or more *external environments*, in addition to a default *language environment*. At each step, the agent either emits a language token or issues an action to an external environment. We formalize this as a partially observed Markov decision process (POMDP) with global state $s_t = (h_t, e_t, z_t)$, where h_t is the token history, $e_t \in \{0, 1, \dots, K\}$ is the active environment ($e_t = 0$ for language), and z_t is the latent state of the external environments. The agent fully observes h_t and e_t , while z_t is only observed indirectly through environment feedback appended to the history.

The agent is a policy $\pi_\theta(a_t | h_t, e_t)$ over actions available in the active environment. Each external environment $i \neq 0$ has an action set $\mathcal{E}_i$ and a transition procedure

$$(o_t, z_{t+1}, \text{exit}) \leftarrow \text{step}_i(h_t, z_t, a_t),$$

which executes $a_t \in \mathcal{E}_i$, returns a token-level observation $o_t \in \mathcal{V}^{(\cdot)}$, updates the latent state, and emits an *exit* flag. The observation is appended to the history, $h_{t+1} \leftarrow h_t \oplus o_t$, and the agent returns to the

language environment when *exit* = true. The active environment yields reward $r_t = R(e_t, h_t, a_t)$, and the agent maximizes $\bar{r} = \sum_t r_t$. Depending on the task, the external environment may serve as a tool supporting language reasoning (e.g., a calculator), or may itself define the task and reward (e.g., sorting hidden numbers via comparisons and swaps; see Figure 2).

Language-only interaction. In existing tool-use and external-interaction approaches (Feng et al., 2025; Singh et al., 2025), agents remain in the language environment: they always select $a_t \in \mathcal{V}$ and extend the history as $h_{t+1} = h_t \oplus a_t$. External interaction is mediated by text patterns, such as `<calculator>...</calculator>` or structured JSON fields, which are parsed into actions in $\mathcal{E}_i$ and routed to the corresponding environment. This makes environment control depend on instruction following and hand-crafted parsers. It also commits the agent to all actions inside a parsed block before receiving intermediate feedback, which can hinder credit assignment in sequential decision-making.

Interaction with expanded action space (ExpA). ExpA lets the agent interact with environments directly by expanding its action space beyond the vocabulary. For each external environment $i \neq 0$, we introduce a *routing action* g_i that transfers control to environment i , together with environment-specific actions $\mathcal{E}_i$. The added action set is

$$\mathcal{E} = \{g_i\}_{i=1}^K \cup \bigcup_{i=1}^K \mathcal{E}_i,$$

and the full action space is $\mathcal{A} = \mathcal{V} \cup \mathcal{E}$. The policy π_θ is defined over $\mathcal{A}$, conditioned on (h_t, e_t) , with only actions valid in the active environment available at each step.

Every rollout begins in the language environment ($e_0 = 0$). The policy may either generate a vocabulary token $a_t \in \mathcal{V}$, extending the history, or select a routing action g_i , which appends its textual description $\text{desc}(g_i)$ and transfers control to environment i . Once inside environment i , the policy selects actions from $\mathcal{E}_i$; step_i then returns an observation, updates the latent state, and decides whether to return control to language. Algorithm 1 gives the details, and Figure 2 illustrates an example.

A key distinction is the direction in which language mediates the interface. ExpA still represents environment observations as text, which are appended to h_t and consumed by the LLM like ordi-

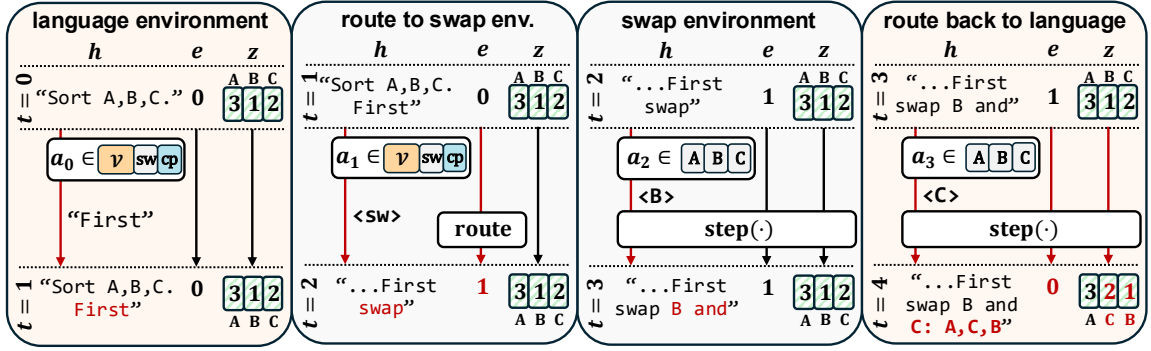


Figure 2: An example rollout with ExpA. Here, $\langle \text{sw} \rangle$ and $\langle \text{cp} \rangle$ route to the swap and compare environments, respectively. Inside them, agents can choose $\langle A \rangle$, $\langle B \rangle$, $\langle C \rangle$ as operands. After two operands are chosen, 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.

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  $a_t \sim \pi_\theta(\cdot \mid h_t, e_t=0)$ 
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  $a_t \sim \pi_\theta(\cdot \mid h_t, e_t \neq 0)$ 
13:     $(o_t, z_{t+1}, \text{exit}) \leftarrow \text{step}_{e_t}(h_t, z_t, a_t)$ 
14:     $h_{t+1} \leftarrow h_t \oplus \text{desc}(a_t) \oplus o_t$ 
15:    if  $\text{exit}=\text{true}$  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  $\tau = \{(h_t, a_t, r_t)\}_{t=0}^{T-1}$ 

```

nary context. However, these observations are not parsed back into actions. What ExpA removes is the reverse direction: the agent’s generated text no longer needs to be parsed to identify tool invocations. Routing and environment actions are direct policy outputs, separating language reasoning from environment control while preserving the standard text-based observation interface.

4 ExpA Reinforcement Learning (EARL)

ExpA gives LLMs explicit actions for interacting beyond language, but these actions are initially unfamiliar to the pretrained model. EARL addresses this with two components: (i) a policy over the expanded action space, initialized from the LLM’s linguistic knowledge, and (ii) *Counterfactual Policy Optimization* (CPO), which encourages exploration

of useful routing actions. CPO optimizes

$$\mathcal{J}_{\text{CPO}}(\theta) = \mathbb{E}_{s_0, \mathcal{T}(s_0)} \left[\frac{1}{m} \sum_{i=1}^m U_i(\mathcal{T}(s_0); \theta) \right], \quad (1)$$

where s_0 is sampled from the training distribution and $\mathcal{T}(s_0) = \{(\tau_i, \tau'_i)\}_{i=1}^m$ contains m factual–counterfactual rollout pairs. Each factual rollout τ_i follows the ExpA rollout procedure, while τ'_i is obtained by forcing a routing action at a plausible intermediate step. The update function U_i uses these pairs to switch between exploration from counterfactual evidence and exploitation of successful factual rollouts.

4.1 A Policy over the Expanded Action Space

Policy parameterization. We extend the standard LLM output head from $|\mathcal{V}|$ vocabulary logits to $|\mathcal{V} \cup \mathcal{E}|$ logits over both tokens and expanded actions. At step t , the encoded representation of h_t is projected to logits, and a softmax is applied only over actions valid in the active environment e_t : $\mathcal{V} \cup \{g_i\}_{i=1}^K$ when $e_t = 0$, and $\mathcal{E}_{e_t}$ otherwise. This yields $\pi_\theta(\cdot \mid h_t, e_t)$ while making the admissible action set explicit.

Policy initialization. Each new action $a \in \mathcal{E}$ has a natural-language description $\text{desc}(a)$, such as an environment name (e.g., “calculator”) or an action label (e.g., “compare”). We initialize the weight of a so that selecting it has approximately the same likelihood as generating 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 environment in which a is admissible. For single-token descriptions, this is done

exactly by copying the pretrained output weight of that token; for multi-token descriptions, we use the mean of their output weights. This gives each new action a linguistic prior and initializes actions independently, which makes adding actions modular.

4.2 Counterfactual Rollouts to Encourage Invoking New Actions

Even with this initialization, useful routing actions may be rarely sampled: a pretrained model has no prior experience selecting g_{calc} as an executable action, even when arithmetic is needed. Standard on-policy gradients struggle in this regime because rarely sampled actions receive little reward-conditioned signal. CPO addresses this by asking what would have happened if the policy had routed into an environment at a plausible point in the trajectory.

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 encourage.
2. Sample an intervention step $t' \in \{t \mid e_t = 0\}$ with probability proportional to $\pi_\theta(\text{desc}(g_i) \mid h_t, e_t=0)$.
3. Copy the factual prefix up to t' .
4. Intervene with $\text{do}(a_{t'} = g_i)$ and apply the ExpA transition.
5. Continue the rollout from $t' + 1$ to obtain τ' .

The intervention distribution uses only the pretrained next-token probability of the routing description. For example, if $\text{desc}(g_{\text{calc}}) = \text{"calculate"}$, positions where "calculate" is linguistically natural receive higher probability. Thus, CPO remains compatible with zero-RL training, without supervised tool-call demonstrations (DeepSeek-AI et al., 2025; Zeng et al., 2025).

4.3 Update Function

For each rollout pair (τ_i, τ'_i) , define

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

where $\bar{r}_i$ and $\bar{r}'_i$ are the cumulative rewards of τ_i and τ'_i , and μ, σ are the mean and standard deviation of $\{\bar{r}_i\}_{i=1}^m$. The function $f(\tau, A; \theta)$ is the standard GRPO/PPO-style update with KL regularization, taking trajectory τ and scalar advantage A as input; the full form is given in Section C.

The first branch is used when all factual rollouts fail, so there is no successful on-policy behavior to

exploit. In this case, the reward difference $\bar{r}'_i - \bar{r}_i$ provides direct evidence about whether forcing the routing action helped. Once at least one factual rollout succeeds, CPO returns to a group-relative advantage over factual rollouts, exploiting strategies already discovered by the policy. Thus, CPO changes the rollout construction and advantage estimator, while leaving the underlying optimizer unchanged.

5 Experiment

5.1 Experimental Setup

We evaluate EARL in controlled settings requiring multi-turn interaction with external environments. Our goal is to isolate two challenges in sequential decision-making: (i) learning reusable interaction policies without disrupting existing language and numerical reasoning, and (ii) adapting actions to hidden state revealed only through interaction. Calc-Bench tests the first challenge through calculator-mediated reasoning, while Sorting tests the second through partial observability. Full data details are provided in Section D.

- **Calc-Bench.** The agent uses a stateless *calculator* environment (Figure 3) and receives exact-match (EM) reward. Calc-Bench tests whether the agent can learn calculator interaction as a reusable policy while preserving arithmetic, language, and numerical understanding. It contains four task types (statistics in Appendix Table 6, examples in Figure 3): (1) *Arithmetic*, which tests out-of-distribution calculator use; (2) *Count-down*, which stresses history-dependent search, where useful operations depend on intermediate outcomes; (3) *GSM8K**, a harder variant of GSM8K (Cobbe et al., 2021) with larger numbers but preserved semantics; and (4) *Count*, which tests whether tool-use training preserves basic numerical understanding.

- **Sorting.** The agent sorts *hidden* numbers by interacting with *compare* and *swap* environments (Figure 3). Comparisons reveal relative order, while swaps modify the hidden state z_t (Figure 2). The reward depends on the final hidden state z_T , with penalties for excessive comparisons and swaps. This task isolates decision-making under partial observability: the agent must gather information, reason over revealed relations, and modify hidden state through environment actions rather than only produce a textual answer. Train-

Example Questions		Environments	
Arithmetic	What is subtraction of 69 divided by 6 plus 7049 plus 13 plus 50 plus 643 from 9936?	Calculator	$+-\times\div=()$ 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**.

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

Table 1: EM results (exact match) on Calc-Bench under joint multitask training. We train each model jointly on all Calc-Bench tasks to assess the benefits of shared representation learning.

ing spans Sort-2 through Sort-5, and evaluation uses Sort-4 and Sort-5.

Baselines. We compare against four paradigms: (1) *SFT+GRPO* (Feng et al., 2025) – supervised fine-tuning on interaction data, then GRPO (Shao et al., 2024); (2) *Prompt+GRPO* (Singh et al., 2025) – interaction patterns provided in the prompt, then GRPO (DeepSeek-AI et al., 2025); (3) *Prompt+CPO* – as Prompt+GRPO, but optimized with our CPO; this isolates the contribution of the expanded action space, since the only difference from EARL is its absence; (4) *Zero-shot* – proprietary models such as GPT-4o (Hurst et al., 2024) without fine-tuning.

Implementation. We use Qwen2.5 (Yang et al., 2024) (base and instruction-tuned, 0.5/3/7B) on NVIDIA A100-80GB GPUs. Following Singh et al. (2025), we use the same hyperparameters and optimization protocol (e.g., KL weight, PPO clipping) across baselines and EARL. Full configurations are in Sections E and F.

5.2 Experimental Results on Calc-Bench

Main results. As shown in Table 1, zero-shot models struggle on Calc-Bench, with GPT-4o reaching only 39.74 overall EM. Training with

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

Table 2: Average occurrence of external interactions, hallucinations and planning phrases in a validation roll-out 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 Table 4.

calculator improves performance, but vocabulary-constrained baselines show uneven gains and remain especially weak on Countdown, where success requires a history-dependent search strategy rather than isolated arithmetic execution. EARL achieves the best overall performance, with up to 10.46 absolute EM gain overall and 17.55 EM gain on Countdown. These gains do not come at the cost of other skills: EARL remains competitive on Arithmetic, Count, and GSM8K* while substantially improving Countdown. The training curves in Figure 4 show the same pattern, suggesting that ExpA learns a reusable interaction policy without disrupting language-based numerical reasoning. We adopt the natural-language tool descrip-

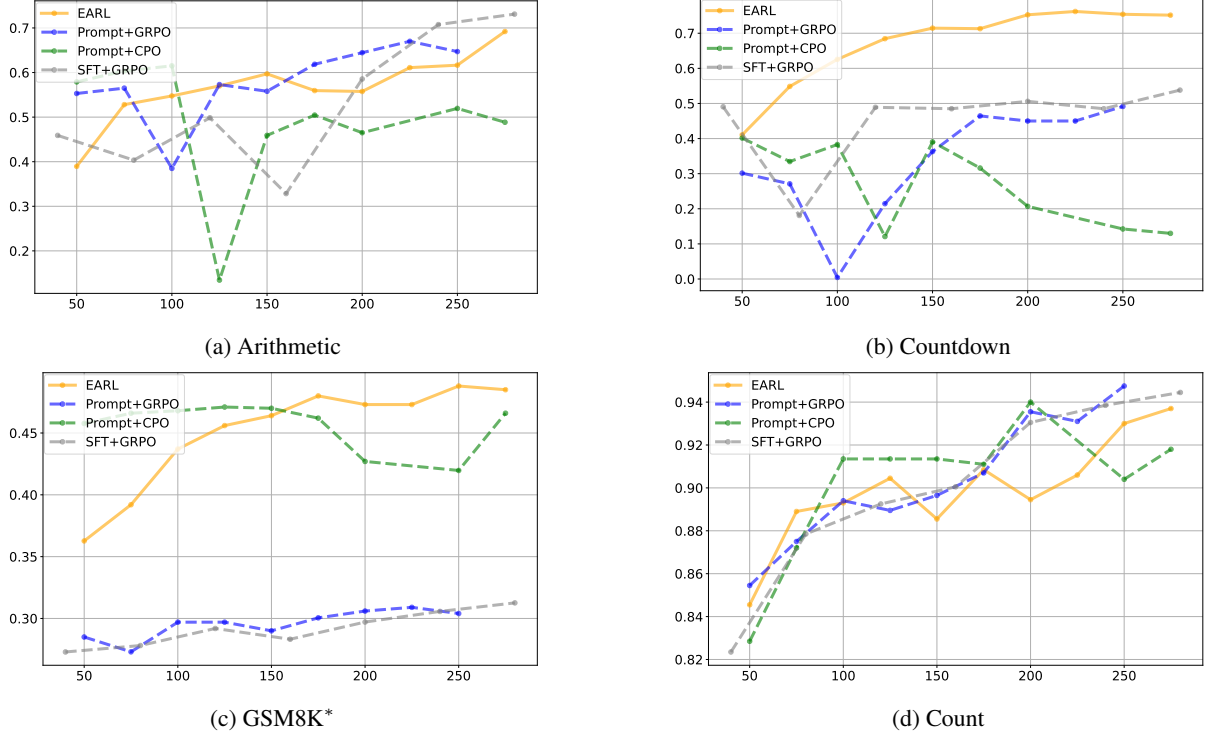


Figure 4: Training dynamics on Calc-Bench under joint multitask RL with a 3B model. Each subplot reports validation exact match (EM) over training steps for one task. EARL improves consistently across tasks and is especially stable on Countdown.

tion as our primary configuration, as it is more general and less dependent on a specific format. Although the `<calculator>...</calculator>` configuration performs slightly better (Appendix Table 9), we attribute this to such special-symbol tags aligning more closely with the base model’s priors for this particular tool interface, rather than to a genuine advantage that would generalize across tools.

Countdown analysis. Table 2 analyzes how models use the calculator on Countdown. Prompt+GRPO invokes the calculator frequently and avoids invalid inputs, but shows little planning behavior, suggesting a tendency toward brute-force trial. Prompt+CPO increases planning cues, but also introduces hallucinated inputs, indicating interference between language reasoning and action learning in the vocabulary-only interface. SFT+GRPO uses the calculator least and often transfers poorly beyond its supervised interaction patterns. In contrast, EARL combines near-zero hallucination with by far the most planning cues, while using a moderate number of interactions. This supports the central hypothesis: separating environment actions from vocabulary generation helps the model learn history-dependent interaction

without disrupting language-based reasoning.

Ablation Study. We ablate the main design choices on Countdown in Table 3. The results show three key patterns.

- *ExpA is the main driver.* Vocabulary-constrained baselines remain much lower even with prompts and constrained decoding (e.g., SFT+GRPO: 62.05). Adding ExpA raises EM to 75.10 even without CPO, and full EARL reaches 80.09.
- *CPO improves exploration within ExpA.* Counterfactual rollouts give consistent additional gains, both with prompts (80.09 vs. 75.10) and without prompts (76.76 vs. 73.79).
- *ExpA reduces reliance on prompts and SFT.* Removing the environment prompt from EARL causes only a modest drop (80.09 to 76.76), and EARL remains strong in the Zero-RL setting (77.31), far above Prompt+GRPO (Zero) at 51.15. This suggests that interaction strategies are learned in the policy rather than rediscovered from prompts.

GSM8K* analysis. In Figure 5, we show the distribution of response lengths among correct rollouts

Method	SFT	CD	Env. Prompt	Zero-RL	CFR	ExpA	EM
SFT-GRPO	✓	✓	✓				62.05
Prompt-GRPO (Zero)		✓	✓	✓			51.15
Prompt-GRPO		✓	✓				58.16
EARL		✓	✓		✓	✓	80.09
– prompt		✓			✓	✓	76.76
– CPO		✓	✓			✓	75.10
– CPO & prompt		✓				✓	73.79
EARL (Zero)		✓	✓	✓	✓	✓	77.31
– prompt		✓		✓	✓	✓	74.56
– CPO		✓	✓	✓		✓	76.45
– CPO & prompt		✓		✓		✓	70.27

Table 3: Countdown ablation study on the contribution of supervised fine-tuning (SFT), constrained decoding (CD), environment prompts, Zero-RL, counterfactual rollouts (CFR), and ExpA. EM denotes exact-match accuracy.

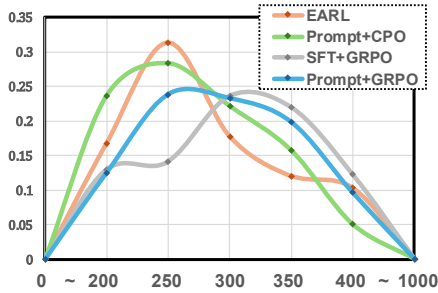


Figure 5: Response token length distribution for correct rollouts in GSM8K*.

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.

Phrase	EARL	Prompt+CPO	Prompt+GRPO	SFT+GRPO
not close	9,961	99	0	18
is close	805	2,930	0	855
close to	3,019	1,138	0	2,253
still close	9	193	0	0
different approach	7,586	92	1,650	15
another approach	0	1,769	0	219
different combination	8,851	39	2,040	377
another combination	1,784	46	0	58
negate	682	4	0	121
far from	8,264	141	0	10
Total	40,961	6,451	3,690	3,926

Table 4: The number of using planning phrases across different training strategies.

Planning phrases analysis. We conduct an in-depth analysis of contingent learning on the Countdown task, where each intermediate response depends on the preceding actions and their outcomes. Table 4 summarizes the results and reveals distinct patterns across methods. For Prompt+GRPO, the

dominant planning phrases are confined to ‘different approach’ and ‘different combination’, indicating a narrow exploration of planning strategies. SFT+GRPO introduces greater diversity in planning phrases, but the increase over Prompt+GRPO is modest (6.4%), suggesting that supervised fine-tuning imposes subjective biases that constrain effective exploration. CPO markedly improves planning diversity, increasing the number of planning phrases by 64.3% and 74.8% over SFT+GRPO and Prompt+GRPO, respectively, enriching the planning process with a broader range of transition options expressed in natural language. Notably, our proposed EARL achieves the highest utilization and diversity of planning phrases. When intermediate responses are incorrect, EARL frequently invokes conditional phrases such as ‘not close’, ‘close to’, and ‘far from’ to guide subsequent steps, enabling the model to either refine the current plan (‘another combination’, ‘different combination’) or initiate a new strategy (‘different approach’). This reflects a more nuanced and adaptive planning behavior than the baselines.

Additional results. We report 0.5B model results in the Section G.1 to demonstrate that it can interact with env as well. We also provide the change of validation performance throughout training on Calc-Bench, as well as case studies that highlight our advantages over previous methods.

5.3 Experimental Results on Sorting

Main results. Figure 6 reports sorting accuracy for Prompt+GRPO, Prompt+CPO, and EARL on Sort-4 and Sort-5, stratified by the minimum number of swaps required. EARL solves Sort-4 perfectly at every difficulty level, while Prompt+CPO degrades as more swaps are required. The gap

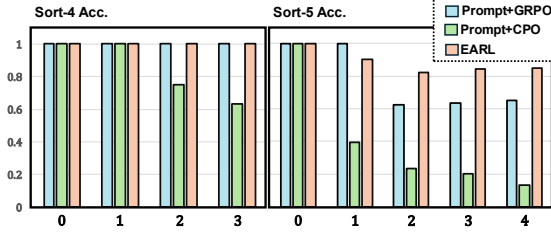


Figure 6: Sort accuracy stratified by the number of required swaps.

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

Table 5: Average number of swaps (SW) and comparisons (CP) to sort 4 random numbers for different algorithms.

widens on Sort-5, where EARL exceeds the strongest baseline (Prompt+GRPO) by up to 21% within a stratum and by more than 10% overall.

Efficiency. We measure the average number of comparisons and swaps EARL uses to sort random Sort-4 inputs. Under greedy decoding, EARL realizes a deterministic decision tree (visualized in Section G.2). Pruning a small number of redundant comparisons yields a simplified variant EARL*, summarized in Algorithm 2: A is used as a pivot, additional comparisons are issued only when needed, and the final ordering is realized by $\text{MIN_SWAP}(\mathcal{R})$ given the accumulated comparison results $\mathcal{R}$. Table 5 compares EARL and EARL* against Prompt+GRPO, classical sorting algorithms, and the theoretical optimum. Both EARL variants match the optimum in swap count and closely approach it in comparison count, outperforming insertion sort and even GCC’s built-in routine.¹

Sorting with RL. Our results connect to recent RL-based algorithm discovery, most notably AlphaDev (Mankowitz et al., 2023). The key distinction is the representation: AlphaDev operates on low-level assembly instructions, while EARL

¹GCC is optimized for processor efficiency (e.g., branch-less `cmove`) rather than minimizing raw operations; “Optimal” denotes the theoretical minimum over the combined number of comparisons and swaps.

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:  $\text{MIN\_SWAP}(\mathcal{R})$ 

```

leverages the LLM’s natural-language vocabulary to represent context and chain reasoning steps. This generality lets EARL transfer pretrained language knowledge into the interactive setting, reaching 100% Sort-4 accuracy in only ~ 70 training steps—several orders of magnitude fewer than AlphaDev’s million-step regime. Sort-5 is not yet perfectly solved; our goal here is to demonstrate that ExpA enables efficient learning in interactive environments, leaving dedicated algorithm discovery and harder settings such as VARSORT for future work.

6 Conclusion

We introduced a new paradigm for Large Language Models (LLMs) to reason with and beyond language when interacting with external environments. Our framework, **ExpA**, augments the policy with routing and environment-specific actions, decoupling reasoning from environment control and removing reliance on text-pattern parsers for action selection. The result is a clean, end-to-end trainable interface in which the agent’s interactions with environments are direct policy outputs, not strings to be parsed. On top of ExpA, we proposed **EARL**, a reinforcement learning method whose counterfactual rollouts provide learning signal for routing actions that a pretrained LLM rarely emits, enabling exploration of the expanded action space without supervised tool-call demonstrations. Empirically, EARL outperforms strong baselines on multi-turn tasks that require environment interaction, with the largest gains on contingent-planning settings where the agent must adapt to intermediate observations. Together, these results establish a modular and principled paradigm for equipping LLMs with explicit interfaces to external environments.

Limitations

The primary objective of this study is to develop a practical method for extending the action space of LLMs, thereby enhancing their reasoning capabilities beyond the intrinsic language knowledge spaces. Our proposed approach, EARL, proves effective in this regard. However, we identify three key limitations. First, due to computational constraints, our empirical evaluation is limited to the Qwen-2.5 model family (up to 7B parameters), and its scaling properties remain to be discovered. Second, we did not investigate optimal action space initialization techniques for novel actions within the continuous learning paradigm. Third, the framework’s performance in complex external environments involving multimodality and diverse action spaces is underexplored. Addressing these limitations to develop a more robust and generalizable solution will be the focus of our future research endeavors.

Acknowledgments

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

- Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. [Augmenting large language models with chemistry tools](#). *Nat. Mac. Intell.*, 6(5):525–535.
- Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. [A survey of monte carlo tree search methods](#). *IEEE Trans. Comput. Intell. AI Games*, 4(1):1–43.
- Yash Chandak, Georgios Theodorou, Chris Nota, and Philip S. Thomas. 2020. [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.
- Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. 2025. [Learning evolving tools for large language models](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *CoRR*, abs/2107.03374.
- Yun-Shiuan Chuang, Agam Goyal, Nikunj Harlalka, Siddharth Suresh, Robert Hawkins, Sijia Yang, Dhavan Shah, Junjie Hu, and Timothy T. Rogers. 2024. [Simulating opinion dynamics with networks of llm-based agents](#). In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3326–3346. Association for Computational Linguistics.
- 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. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- 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, and 81 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Gregory Farquhar, Laura Gustafson, Zeming Lin, Shimon Whiteson, Nicolas Usunier, and Gabriel Synnaeve. 2020. [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.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. [Retool: Reinforcement learning for strategic tool use in llms](#). *CoRR*, abs/2504.11536.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [PAL: program-aided language models](#). In *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.

- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujia Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. [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.
- Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. [Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings](#). In *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*.
- Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. [Solving math word problems by combining language models with symbolic solvers](#). *CoRR*, abs/2304.09102.
- 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, and 79 others. 2024. [Gpt-4o system card](#). *CoRR*, abs/2410.21276.
- Ayush Jain, Norio Kosaka, Kyung-Min Kim, and Joseph J. Lim. 2022. [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.
- Ayush Jain, Andrew Szot, and Joseph J. Lim. 2020. [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.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. [Swe-bench: Can language models resolve real-world github issues?](#) In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. [Planning and acting in partially observable stochastic domains](#). *Artif. Intell.*, 101(1-2):99–134.
- 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 Tenenholz. 2022. [MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning](#). *CoRR*, abs/2205.00445.
- Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2022. [Towards continual reinforcement learning: A review and perspectives](#). *J. Artif. Intell. Res.*, 75:1401–1476.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. [Agentbench: Evaluating llms as agents](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Zhaoyang Liu, Yinan He, Wenhai Wang, Weiyun Wang, Yi Wang, Shoufa Chen, Qinglong Zhang, Zeqiang Lai, Yang Yang, Qingyun Li, Jiashuo Yu, Kunchang Li, Zhe Chen, Xue Yang, Xizhou Zhu, Yali Wang, Limin Wang, Ping Luo, Jifeng Dai, and Yu Qiao. 2023. [Interngpt: Solving vision-centric tasks by interacting with chatbots beyond language](#). *CoRR*, abs/2305.05662.
- Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. [Octotools: An agentic framework with extensible tools for complex reasoning](#). *CoRR*, abs/2502.11271.
- Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, and 11 others. 2023. [Faster sorting algorithms discovered using deep reinforcement learning](#). *Nat.*, 618(7964):257–263.
- Amogh Mannekote, Adam Davies, Jina Kang, and Kristy Elizabeth Boyer. 2025. [Can llms reliably simulate human learner actions? A simulation authoring framework for open-ended learning environments](#). In *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.
- Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. [TALM: tool augmented language models](#). *CoRR*, abs/2205.12255.
- 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, and 24 others. 2025. [Tool learning with foundation models](#). *ACM Comput. Surv.*, 57(4):101:1–101:40.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *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*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient RLHF framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflection: language agents with verbal reinforcement learning](#). In *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*.
- Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. [Agentic reasoning and tool integration for llms via reinforcement learning](#). *CoRR*, abs/2505.01441.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. [Trial and error: Exploration-based trajectory optimization for LLM agents](#). *CoRR*, abs/2403.02502.
- Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press.
- Andrew Szot, Bogdan Mazouze, Omar Attia, Aleksei Timofeev, Harsh Agrawal, R. Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. 2025. [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.
- Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. [Mathscale: Scaling instruction tuning for mathematical reasoning](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Jingwei Wang, Zai Zhang, Hao Qian, Chunjing Gan, Binbin Hu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Bin Shi, and Bo Dong. 2025a. [Enhancing LLM tool use with high-quality instruction data from knowledge graph](#). *CoRR*, abs/2506.21071.
- Qian Wang, Jiaying Wu, Zhenheng Tang, Bingqiao Luo, Nuo Chen, Wei Chen, and Bingsheng He. 2025b. [What limits llm-based human simulation: Llms or our design?](#) *CoRR*, abs/2501.08579.
- 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, and 22 others. 2024. [Qwen2.5 technical report](#). *CoRR*, abs/2412.15115.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [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.
- Weihaio Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. [Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild](#). *CoRR*, abs/2503.18892.
- Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. [Evaluating and improving tool-augmented computation-intensive math reasoning](#). In *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*.
- Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. [Webarena: A realistic web environment for building autonomous agents](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Contents of Appendix

A	Usage of AI Assistant	14
B	Policy on expanded action space	14
C	Update Rule	14
D	Dataset Details	14
E	Baseline Details	15
F	Implementation Details	15
F.1	Calc-Bench Implementation Details	15
F.2	Sorting Implementation Details . .	18
F.3	Prompts	21
G	Additional Experimental Results	23
G.1	Calc-Bench	23
G.2	Sorting	23
H	Additional Analysis	24
H.1	Action Initialization Analysis . . .	24
I	Case Study	26

A Usage of AI Assistant

We use AI assistants or tools such as ChatGPT and Grammarly to correct grammar errors and polish the language.

B Policy on expanded action space

Our parameterization of the π_θ uses a linear-softmax layer applied to an encoding $g(h_t) \in \mathbb{R}^d$ of the history (context) h_t by the LLM. The layer uses a weight matrix $W = [w_1, \dots, w_N]^\top$, where $N = |\mathcal{E}|$ is the total number of actions across all environments, including the language one, and $w_a \in \mathbb{R}^d$ are the parameters used to compute the logit of action a for $a \in [N]$. The softmax is restricted to actions that are available in the active environment e_t . We define, for environments $e \in \{0, 1, \dots, K\}$,

$$\sigma_e(z)_a = \frac{e^{z_a}}{\sum_{a' \in \tilde{\mathcal{E}}_e} e^{z_{a'}}} \mathbb{1}[a \in \tilde{\mathcal{E}}_e].$$

Here, $\tilde{\mathcal{E}}_0 = \{g_e\}_{e \in [K]} \cup \mathcal{E}_0$ and $\tilde{\mathcal{E}}_e = \mathcal{E}_e$ for $e = 1, \dots, K$. With this,

$$\pi_\theta(\cdot \mid h_t, e_t) = \sigma_{e_t}(Wg(h_t)).$$

C 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 Equation (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 $\mathcal{V}$.

D 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.

- *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 num-

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

Table 6: Statistics of the Calc-Bench datasets. Language portions refer to the portion of questions where operations or numbers are written in natural language.

ber of specific items). The training set includes 1,000 examples and the test set includes 2,000.

Sorting Dataset Details. 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.

E Baseline Details

In our experiments, we employ open-source LLM Qwen2.5 (Yang et al., 2024) as the backbone, which has been designed to address a wide range of applications, including coding and mathematics. We select both base and instruction-tuned variants with model sizes of 0.5B, 3B, and 7B parameters for our experiments.

Here we provide details of each baseline methods as follows:

- **GPT-4o** (Hurst et al., 2024) is a frontier large language model developed by OpenAI, demonstrating advanced capabilities in reasoning with various contexts. We use standard prompt instruction as the key to generating a response.

- **Prompt-based RL** is a cost-effective and effective method for LLMs, specifically designed to train them by optimizing their responses to given prompts. We employ the cutting-edge approach Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for our **SFT+GRPO** and **Prompt-GRPO** (DeepSeek-AI et al., 2025) baselines. We replace GRPO with our proposed CPO as **Prompt-CPO** baseline.

SFT Data Curation. We follow (Tang et al., 2024), leveraging frontier LLM GPT-4o to generate solutions on the Countdown task. In order to obtain high-quality SFT data, we only select solutions that GPT-4o can correctly answer with no more than 1K tokens.

F Implementation Details

Our implementation of EARL efficiently supports ExpA rollouts through a customized vLLM backend (Kwon et al., 2023) and integration with the VeRL training library (Sheng et al., 2025). We follow the configurations outlined in VeRL. To ensure the reproducibility of our findings, detailed implementation instructions are provided below.

F.1 Calc-Bench Implementation Details

Prompt+GRPO Training Config <Qwen-2.5-3B-Instruct>

```
apptainer exec --nv envs/verl.sif bash -c "  
  ray start --head --port=6383 &&  
  set -x &&  
  python3 -m verl.trainer.main_earl  
  actor_rollout_ref.earl.model.freeze_base_model=False  
  actor_rollout_ref.earl.model.init_from_base=True  
  actor_rollout_ref.earl.training.tools=['calculator']  
  algorithm.adv_estimator=grpo  
  tool_config_path=tool_configs/combined_calculator_baseline.yaml  
  data.train_files=./data/calc_bench/combined_baseline/train.parquet  
  data.val_files=./data/calc_bench/combined_baseline/test.parquet  
  data.train_batch_size=256  
  data.max_prompt_length=384  
  data.max_response_length=1024  
  data.filter_overlong_prompts=True  
  data.truncation='error'  
  actor_rollout_ref.model.path=Qwen/Qwen2.5-3B-Instruct  
  actor_rollout_ref.actor.optim.lr=1e-6  
  actor_rollout_ref.model.use_remove_padding=True  
  actor_rollout_ref.actor.ppo_mini_batch_size=32  
  actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8  
  actor_rollout_ref.actor.use_kl_loss=True  
  actor_rollout_ref.actor.kl_loss_coef=0.001  
  actor_rollout_ref.actor.kl_loss_type=low_var_kl  
  actor_rollout_ref.actor.entropy_coeff=0  
  actor_rollout_ref.model.enable_gradient_checkpointing=True  
  actor_rollout_ref.actor.entropy_from_logits_with_chunking=True  
  actor_rollout_ref.actor.fsdp_config.param_offload=False  
  actor_rollout_ref.actor.fsdp_config.optimizer_offload=False  
  actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8  
  actor_rollout_ref.rollout.tensor_model_parallel_size=1  
  actor_rollout_ref.rollout.name=vllm  
  actor_rollout_ref.rollout.gpu_memory_utilization=0.6  
  actor_rollout_ref.rollout.n=8  
  actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=8  
  actor_rollout_ref.ref.fsdp_config.param_offload=True  
  algorithm.use_kl_in_reward=False  
  trainer.critic_warmup=0  
  trainer.logger=['console', 'tensorboard']  
  trainer.project_name='earl'  
  trainer.experiment_name='calc_bench/qwen2.5-3b/prompt-grpo'  
  trainer.validation_data_dir=./val_result/calc_bench/3b/prompt-grpo  
  trainer.n_gpus_per_node=4  
  trainer.nnodes=1  
  trainer.save_freq=25  
  trainer.test_freq=25  
  trainer.total_epochs=4
```

EARL Training Config <Qwen-2.5-3B-Instruct>

```
apptainer exec --nv envs/verl.sif bash -c "  
  ray start --head --port=6383 &&  
  set -x &&  
  python3 -m verl.trainer.main_earl  
  actor_rollout_ref.earl.model.freeze_base_model=False  
  actor_rollout_ref.earl.model.init_from_base=True  
  actor_rollout_ref.earl.training.tools=['calculator']  
  algorithm.adv_estimator=trpo  
  tool_config_path=tool_configs/combined_calculator.yaml  
  data.train_files=./data/calc_bench/combined_earl/train.parquet  
  data.val_files=./data/calc_bench/combined_earl/test.parquet  
  data.train_batch_size=256  
  data.max_prompt_length=384  
  data.max_response_length=1024  
  data.filter_overlong_prompts=True  
  data.truncation='error'  
  actor_rollout_ref.model.path=Qwen/Qwen2.5-3B-Instruct  
  actor_rollout_ref.actor.optim.lr=1e-6
```

```

actor_rollout_ref.model.use_remove_padding=True
actor_rollout_ref.actor.ppo_mini_batch_size=32
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8
actor_rollout_ref.actor.use_kl_loss=True
actor_rollout_ref.actor.kl_loss_coef=0.001
actor_rollout_ref.actor.kl_loss_type=low_var_kl
actor_rollout_ref.actor.entropy_coeff=0
actor_rollout_ref.model.enable_gradient_checkpointing=True
actor_rollout_ref.actor.entropy_from_logits_with_chunking=True
actor_rollout_ref.actor.fsdps_config.param_offload=False
actor_rollout_ref.actor.fsdps_config.optimizer_offload=False
actor_rollout_ref.rollout.max_num_batched_tokens=7200
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.rollout.tensor_model_parallel_size=1
actor_rollout_ref.rollout.name=vllm
actor_rollout_ref.rollout.gpu_memory_utilization=0.6
actor_rollout_ref.rollout.n=8
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.ref.fsdps_config.param_offload=True
algorithm.use_kl_in_reward=False
trainer.critic_warmup=0
trainer.logger=['console', 'tensorboard']
trainer.project_name='earl'
trainer.experiment_name='calc_bench/qwen2.5-3b/earl-trpo'
trainer.validation_data_dir='./val_result/calc_bench/3b/earl'
trainer.n_gpus_per_node=4
trainer.nnodes=1
trainer.save_freq=25
trainer.test_freq=25
trainer.total_epochs=4

```

Prompt+GRPO Training Config <Qwen-2.5-7B-Instruct>

```

apptainer exec --nv envs/verl.sif bash -c "
ray start --head --port=6384 &&
set -x &&
python3 -m verl.trainer.main_earl
actor_rollout_ref.earl.model.freeze_base_model=False
actor_rollout_ref.earl.model.init_from_base=True
actor_rollout_ref.earl.training.tools=['calculator']
algorithm.adv_estimator=grpo
tool_config_path=tool_configs/combined_calculator_baseline.yaml
data.train_files=./data/calc_bench/combined_baseline/train.parquet
data.val_files=./data/calc_bench/combined_baseline/test.parquet
data.train_batch_size=256
data.max_prompt_length=384
data.max_response_length=1024
data.filter_overlong_prompts=True
data.truncation='error'
actor_rollout_ref.model.path=Qwen/Qwen2.5-7B-Instruct
actor_rollout_ref.actor.optim.lr=1e-6
actor_rollout_ref.model.use_remove_padding=True
actor_rollout_ref.actor.ppo_mini_batch_size=32
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8
actor_rollout_ref.actor.use_kl_loss=True
actor_rollout_ref.actor.kl_loss_coef=0.001
actor_rollout_ref.actor.kl_loss_type=low_var_kl
actor_rollout_ref.actor.entropy_coeff=0
actor_rollout_ref.model.enable_gradient_checkpointing=True
actor_rollout_ref.actor.entropy_checkpointing=True
actor_rollout_ref.actor.entropy_from_logits_with_chunking=True
actor_rollout_ref.actor.fsdps_config.param_offload=False
actor_rollout_ref.actor.fsdps_config.optimizer_offload=False
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.rollout.tensor_model_parallel_size=1
actor_rollout_ref.rollout.name=vllm
actor_rollout_ref.rollout.gpu_memory_utilization=0.6
actor_rollout_ref.rollout.n=8
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=32

```

```

actor_rollout_ref.ref.fsdps_config.param_offload=True
algorithm.use_kl_in_reward=False
trainer.critic_warmup=0
trainer.logger=['console','tensorboard']
trainer.project_name='earl'
trainer.experiment_name='calc_bench/qwen2.5-7b/prompt-grpo'
trainer.validation_data_dir='./val_result/calc_bench/7b/prompt-grpo'
trainer.n_gpus_per_node=4
trainer.nnodes=1
trainer.save_freq=100
trainer.test_freq=10
trainer.total_epochs=4

```

EARL Training Config <Qwen-2.5-7B-Instruct>

```

apptainer exec --nv envs/verl.sif bash -c "
ray start --head --port=6384 &&
set -x &&
python3 -m verl.trainer.main_earl
actor_rollout_ref.earl.model.freeze_base_model=False
actor_rollout_ref.earl.model.init_from_base=True
actor_rollout_ref.earl.training.tools=['calculator']
algorithm.adv_estimator=trpo
tool_config_path=tool_configs/combined_calculator.yaml
data.train_files=./data/calc_bench/combined_earl/train.parquet
data.val_files=./data/calc_bench/combined_earl/test.parquet
data.train_batch_size=256
data.max_prompt_length=384
data.max_response_length=1024
data.filter_overlong_prompts=True
data.truncation='error'
actor_rollout_ref.model.path=Qwen/Qwen2.5-7B-Instruct
actor_rollout_ref.actor.optim.lr=1e-6
actor_rollout_ref.model.use_remove_padding=True
actor_rollout_ref.actor.ppo_mini_batch_size=32
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8
actor_rollout_ref.actor.use_kl_loss=True
actor_rollout_ref.actor.kl_loss_coef=0.001
actor_rollout_ref.actor.kl_loss_type=low_var_kl
actor_rollout_ref.actor.entropy_coeff=0
actor_rollout_ref.model.enable_gradient_checkpointing=True
actor_rollout_ref.actor.entropy_checkpointing=True
actor_rollout_ref.actor.entropy_from_logits_with_chunking=True
actor_rollout_ref.actor.fsdps_config.param_offload=False
actor_rollout_ref.actor.fsdps_config.optimizer_offload=False
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.rollout.tensor_model_parallel_size=1
actor_rollout_ref.rollout.name=vllm
actor_rollout_ref.rollout.gpu_memory_utilization=0.6
actor_rollout_ref.rollout.n=8
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=32
actor_rollout_ref.ref.fsdps_config.param_offload=True
algorithm.use_kl_in_reward=False
trainer.critic_warmup=0
trainer.logger=['console','tensorboard']
trainer.project_name='earl'
trainer.experiment_name='calc_bench/qwen2.5-7b/earl-trpo'
trainer.validation_data_dir='./val_result/calc_bench/7b/earl'
trainer.n_gpus_per_node=4
trainer.nnodes=1
trainer.save_freq=100
trainer.test_freq=10
trainer.total_epochs=4

```

F.2 Sorting Implementation Details

Prompt+GRPO Training Config <Qwen-2.5-7B-Instruct>

```

apptainer exec --nv envs/verl.sif bash -c "
ray start --head --port=6384 &&

```

```

set -x &&
python3 -m verl.trainer.main_earl
actor_rollout_ref.earl.model.freeze_base_model=False
actor_rollout_ref.earl.model.init_from_base=True
actor_rollout_ref.earl.training.tools=['swap', 'compare']
algorithm.adv_estimator=grpo
tool_config_path=tool_configs/sorting_baseline.yaml
data.train_files=./data/sort_baseline_train/train.parquet
data.val_files=./data/sort_earl_4_5/test.parquet
data.train_batch_size=256
data.max_prompt_length=384
data.max_response_length=384
data.filter_overlong_prompts=True
data.truncation='error'
actor_rollout_ref.model.path=Qwen/Qwen2.5-7B-Instruct
actor_rollout_ref.actor.optim.lr=1e-6
actor_rollout_ref.model.use_remove_padding=True
actor_rollout_ref.actor.ppo_mini_batch_size=32
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8
actor_rollout_ref.actor.use_kl_loss=True
actor_rollout_ref.actor.kl_loss_coef=0.001
actor_rollout_ref.actor.kl_loss_type=low_var_kl
actor_rollout_ref.actor.entropy_coeff=0
actor_rollout_ref.model.enable_gradient_checkpointing=True
actor_rollout_ref.actor.entropy_checkpointing=True
actor_rollout_ref.actor.entropy_from_logits_with_chunking=True
actor_rollout_ref.actor.fsdps_config.param_offload=False
actor_rollout_ref.actor.fsdps_config.optimizer_offload=False
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.rollout.tensor_model_parallel_size=1
actor_rollout_ref.rollout.name=vllm
actor_rollout_ref.rollout.gpu_memory_utilization=0.6
actor_rollout_ref.rollout.n=8
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.ref.fsdps_config.param_offload=True
algorithm.use_kl_in_reward=False
trainer.critic_warmup=0
trainer.logger=['console', 'tensorboard']
trainer.project_name='earl'
trainer.experiment_name='sorting/qwen2.5-7b/prompt-grpo-sort'
trainer.validation_data_dir=./val_result/sorting/7b/prompt-grpo
trainer.n_gpus_per_node=4
trainer.nnodes=1
trainer.save_freq=5000
trainer.test_freq=3
trainer.total_epochs=1

```

EARL Training Config <Qwen-2.5-7B-Instruct>

```

apptainer exec --nv envs/verl.sif bash -c "
ray start --head --port=6384 &&
set -x &&
python3 -m verl.trainer.main_earl
actor_rollout_ref.earl.model.freeze_base_model=False
actor_rollout_ref.earl.model.init_from_base=True
actor_rollout_ref.earl.training.tools=['swap', 'compare']
algorithm.adv_estimator=trpo
tool_config_path=tool_configs/sorting.yaml
data.train_files=./data/sort_earl_train/train.parquet
data.val_files=./data/sort_earl_4_5/test.parquet
data.train_batch_size=256
data.max_prompt_length=384
data.max_response_length=384
data.filter_overlong_prompts=True
data.truncation='error'
actor_rollout_ref.model.path=Qwen/Qwen2.5-7B-Instruct
actor_rollout_ref.actor.optim.lr=1e-6
actor_rollout_ref.model.use_remove_padding=True
actor_rollout_ref.actor.ppo_mini_batch_size=32

```

```
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=8
actor_rollout_ref.actor.use_kl_loss=True
actor_rollout_ref.actor.kl_loss_coef=0.001
actor_rollout_ref.actor.kl_loss_type=low_var_kl
actor_rollout_ref.actor.entropy_coeff=0
actor_rollout_ref.model.enable_gradient_checkpointing=True
actor_rollout_ref.actor.entropy_checkpointing=True
actor_rollout_ref.actor.entropy_from_logits_with_chunking=True
actor_rollout_ref.actor.fsdps_config.param_offload=False
actor_rollout_ref.actor.fsdps_config.optimizer_offload=False
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.rollout.tensor_model_parallel_size=1
actor_rollout_ref.rollout.name=vllm
actor_rollout_ref.rollout.gpu_memory_utilization=0.6
actor_rollout_ref.rollout.n=8
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=8
actor_rollout_ref.ref.fsdps_config.param_offload=True
algorithm.use_kl_in_reward=False
trainer.critic_warmup=0
trainer.logger=['console','tensorboard']
trainer.project_name='earl'
trainer.experiment_name='sorting/qwen2.5-7b/earl-sort'
trainer.validation_data_dir='./val_result/sorting/7b/earl'
trainer.n_gpus_per_node=4
trainer.nnodes=1
trainer.save_freq=5000
trainer.test_freq=3
trainer.total_epochs=1
```

E.3 Prompts

We highlight the **environment prompt**, **question**, and **task** in our instruction prompt.

Prompt Example <Calc-Bench: Arithmetic>

<system>:

You are a helpful assistant. You first think about the reasoning process in the mind and then provide the user with the answer.

You are allowed to use calculator by wrapping the expression with `<calculator>` `</calculator>` tags. The calculator output will follow inside `<result>` `</result>` tags. For example, `<calculator> (12 + 6) / 3 </calculator>` will produce `<result> 6 </result>`. You can use calculator multiple times in your reasoning process.

<user>:

What is $7 + (4 * 366 + 32287 - 7471)$? Output your answer after `####`.

<assistant>:

Let me solve this step by step.

Prompt Example <Calc-Bench: Countdown>

<system>:

You are a helpful assistant. You first think about the reasoning process in the mind and then provide the user with the answer.

You are allowed to use calculator with the `'calculate'` keyword. For example, you may calculate $(12 + 6) / 3 = 6$. You can use calculator multiple times in your reasoning process.

<user>:

Using the numbers [3697, 5655, 1199, 11], create an equation that equals 74587492. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once. Output the final answer after `####`. For example, given numbers [1, 2, 3] and target number 1, output #### $(1 + 2) / 3$.

<assistant>:

Let me solve this step by step.

Prompt Example <Calc-Bench: Count>

<system>:

You are a helpful assistant. You first think about the reasoning process in the mind and then provide the user with the answer.

<user>:

How many times does the digit 2 appear in the number eighty-three million, seven hundred and forty-five thousand and thirty-nine? Output your answer after `####`. For example, given the number 121 and digit 1, output #### 2.

<assistant>:

Let me solve this step by step.

Prompt Example <Calc-Bench: GSM8K*>

<system>:

You are a helpful assistant. You first think about the reasoning process in the mind and then provide the user with the answer.

<user>:

Natalia sold clips to 40770 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? Output your final answer after `####`.

<assistant>:

Let me solve this step by step.

Prompt Example <Sorting>

<system>:

You are a helpful assistant. You first think about the reasoning process in the mind and then provide the user with the answer.

You have access to the following tools:

- compare tool: compare A and B: $A > B$
- swap tool: swap A and B $\Rightarrow$ B, A

<user>:

Sort the following items in descending order: A, B, C, D, E. While you do not know the values of the items, you can compare any two items using the compare tool. Once you know the order of them, you can use the swap tool multiple times to complete the task. For example, to sort A, B in descending order, if you find $A < B$ with the compare tool, you can use the swap tool on A and B to complete the task. Use fewest possible comparisons and swaps to complete the task. Stop when the sequence is sorted and do not output any answer.

<assistant>:

Let me solve this step by step.

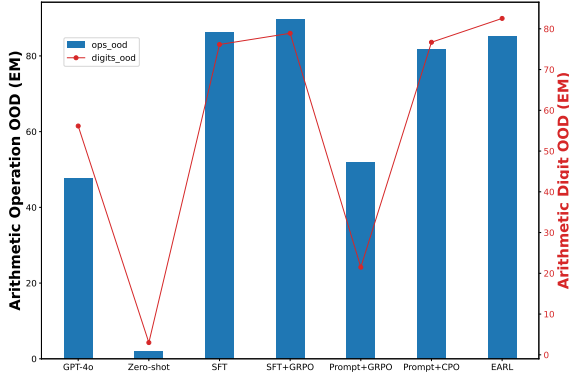


Figure 7: Performance comparison on the Calc-Bench Arithmetic. Arithmetic Operation Out-of-Distribution (OOD) is shown as bars, and Arithmetic Digit OOD as lines. Both plots correspond to the lightweight model Qwen-2.5-0.5B-Instruct.

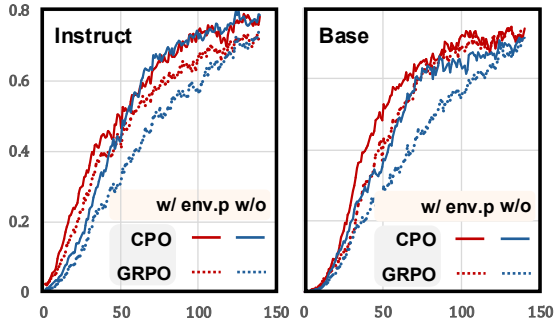


Figure 8: ExpA training reward vs. iteration.

G Additional Experimental Results

G.1 Calc-Bench

Evaluation on Lightweight Model. Experimental results in Figure 7 demonstrate that our EARL can effectively improve the Arithmetic task for a lightweight model with only 0.5 billion parameters (Qwen-2.5-0.5B-Instruct). In this experiment, as described in Table 7, we consider two Out-of-Distribution (OOD) variants, including *Operation OOD* and *Digit OOD*. Due to the limitation of the model’s capability, we observe that Qwen-2.5-0.5B-Instruct has almost zero inherent knowledge to handle arithmetic tasks without fine-tuning. With enough training resources, SFT can achieve a significant performance gain for both OOD settings. Notably, EARL can effectively benefit the lightweight model, achieving comparable performance on the *Operation OOD* task, and even better results on the *Digit OOD* task.

Evaluation on Collective Tool Learning. In Figure 4, we compare EARL with three baseline meth-

ods, where the tool learning trajectory of EARL is represented in yellow. We observed that all the methods in Figure 4d show a similar trend, demonstrating a distinct learning behavior with respect to other tasks. One possible reason is that this task is relatively easy, which could be well addressed by the model’s inherent language knowledge spaces; introducing external action spaces won’t enhance this task much. For the other three tasks, we observe that EARL reveals the efficiency and effectiveness of learning, indicating that all tasks are complementary to achieve enhanced and robust math reasoning with a better trade-off. Our findings suggest that EARL is a practical and scalable framework to expand action spaces.

Additional Ablation Study. To validate the robustness of EARL on different environment configurations, we conduct a comprehensive ablation study, shown in Table 9. All experimental results and findings validate that our proposed EARL achieves the best robustness, indicating that EARL is an effective framework with the potential to handle diverse external environments. Furthermore, by eliminating the need for environment-specific configurations, EARL develops a generalized understanding of tool interactions that is more naturally aligned with language.

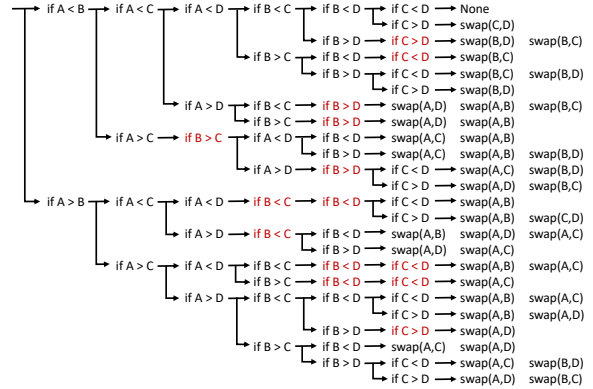


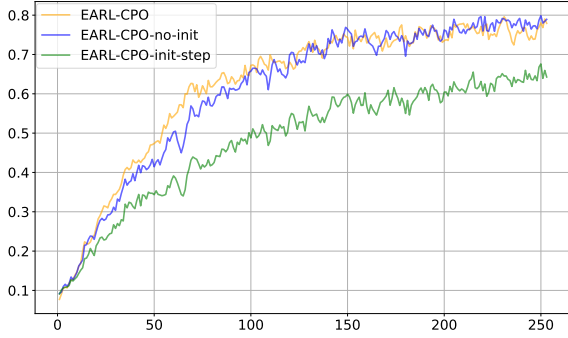
Figure 10: Decision tree induced by EARL on Sort-4. Red nodes indicate redundant comparisons that can be pruned to obtain EARL*.

G.2 Sorting

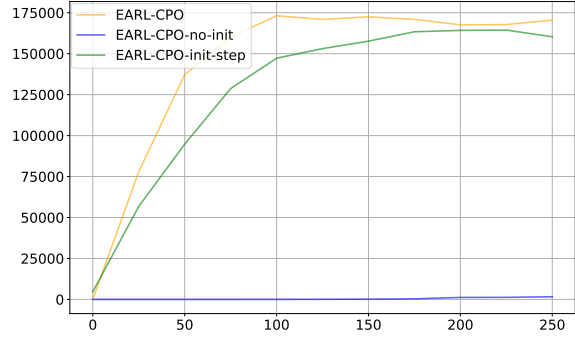
Figure 10 visualizes the deterministic decision tree induced by EARL on Sort-4 inputs under greedy decoding. Each internal node represents a binary comparison, and each leaf corresponds to a sequence of swaps for producing a sorted output. Red nodes indicate redundant comparisons that do not affect the final decision and can be pruned to yield the

Task	Max number (10^x)		#Operands		Lang. portion		#Instances	
	Train	Test	Train	Test	Train	Test	Train	Test
Operation OOD	4	4	3	6	5%	70%	20,000	2,000
Digit OOD	3	4	5	5	NA	NA	20,000	2,000

Table 7: Statistics of the Arithmetic OOD setting. Language portions refer to the portion of questions where operations or numbers are written in natural language.



(a) Training reward.



(b) Tool use count.

Figure 9: Comparison of training reward and tool use count under different initialization strategies.

	Instruct		Base	
	w/ env.p	w/o	w/ env.p	w/o
ExpA+CPO (EARL)	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	-

Table 8: 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.

exhibits some tool use but remains less effective than full initialization in both tool utilization and reward.

simplified variant EARL* described in the main text.

H Additional Analysis

H.1 Action Initialization Analysis

In Section 4.1 we described our policy parameterization and initialization strategy for expanded actions. Here we provide additional empirical evidence supporting this design. As shown in Figure 9, EARL-CPO with full initialization, rapidly learn to invoke tools and achieve the highest rewards. In contrast, models trained with EARL-CPO-no-init almost never use tools and instead converge to a suboptimal language-only strategy. We also evaluate a partial variant, EARL-CPO-init-step, in which only calculator button actions are initialized from their natural language descriptions, and find that it

Method	Configuration	Calc-Bench				
		Arithmetic	Countdown	Count	GSM8K*	Overall
EARL	_calculate	69.20	75.15	93.70	48.53	71.65
	<calculator></..>	75.80	78.00	92.00	51.12	74.23
Prompt+GRPO	_<calculate>_	50.05	34.95	82.25	27.46	48.68
	_<calculate>	64.70	49.15	94.75	30.39	59.75
Prompt+CPO	_<calculate>_	63.05	60.15	88.70	46.63	64.63
	_<calculate>	61.50	38.30	91.35	46.80	59.49

Table 9: Ablation EM results of various environment configurations.

I Case Study

Below are examples from the Countdown task under different methods.

EARL

<question>:

Using the numbers [7, 681, 8, 6348], create an equation that equals -5666.125. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<EARL>:

We need to create an equation using the numbers 7, 681, 8, and 6348 that equals -5666.125

One approach is to try different combinations of operations and see if we can get close to the target.

Let's calculate $6348 - 681 - 7 - 8 = 5652$

This is not close to -5666.125. Let's try another combination.

Let's calculate $6348 - 681 + 7 - 8 = 5666$

We need to negate this to get -5666. Let's try negating it.

So, $-(6348 - 681 + 7 - 8) = -5666$

This is close, but not exactly -5666.125.

Let's try a different approach.

Let's calculate $6348 - 681 - 7 + 8 = 5668$

This is not close either. Let's try a different combination.

Let's calculate $6348 - 681 + 7 + 8 = 5682$

This is still not close. Let's try a different approach.

Let's calculate $6348 - 681 - 7 / 8 = 5666.125$

We need to negate this to get -5666.125.

So, $-(6348 - 681 - 7 / 8) = -5666.125$

This gives us the correct result.

Therefore, the final answer is:#### $-(6348 - 681 - 7 / 8)$

<reward>:1.0

<question>:

Using the numbers [3, 8890, 6, 1], create an equation that equals 8888. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<EARL>:

We need to create an equation using the numbers 3, 8890, 6, and 1 that equals 8888.

One approach is to try different combinations of operations and see if we can get close to 8888.

Let's calculate $8890 - 8888 = 2$

This isn't close to 8888. Let's try another combination.

Let's calculate $8890 - 6 - 1 + 3 = 8886$

This is still not quite 8888. Let's try a different approach.

Let's calculate $8890 - 6 + 1 + 3 = 8888$

This gives us the correct result.

Therefore, the equation is: $8890 - 6 + 1 + 3 = 8888$

$(8890 - 6 + 1 + 3)$

<reward>:1.0

Prompt+CP0

<question>:

Using the numbers [7, 681, 8, 6348], create an equation that equals

-5666.125. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<Prompt+CP0>:

First, I'll consider the numbers [7, 681, 8, 6348] and try to form an equation that equals -5666.125.

One approach could be to use subtraction and division to get close to -5666.125.

Let's try: `<calculator> (6348 - 681) - 8 - 7 </calculator>`

`<result> 5652 </result>` This is close but positive.

Another approach could be:

`<calculator> (6348 - 681) - 8 + 7 </calculator>`

`<result> 5666 </result>` This is close but positive.

Let's try: `<calculator> (6348 - 681) + 8 - 7 </calculator>`

`<result> 5668 </result>` This is close but positive.

Let's try: `<calculator> (6348 - 681) + 8 + 7 </calculator>`

`<result> 5682 </result>` This is close but positive.

Let's try: `<calculator> (681 - 6348) - 8 - 7 </calculator>`

`<result> -5682 </result>` This is close but positive.

Let's try: `<calculator> (681 - 6348) - 8 + 7 </calculator>`

`<result> -5668 </result>` This is close but positive.

Let's try: `<calculator> (681 - 6348) + 8 - 7 </calculator>`

`<result> -5666 </result>` This is the target number.

Therefore, the final answer is: ##### (681 - 6348) + 8 - 7

<reward>:0.0

<question>:

Using the numbers [3, 8890, 6, 1], create an equation that equals 8888. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<Prompt+CP0>:

First, I'll consider the numbers [3, 8890, 6, 1]

and try to form an equation that equals 8888.

One approach could be to use the largest numbers

first to get close to 8888.

Let's try: `<calculator> (8890 - 6) - 3 - 1 </calculator>`

`<result> 8880 </result>` This is close but slightly lower.

Another approach could be: `<calculator> (8890 - 6) - 3 + 1 </calculator>`

`<result> 8882 </result>`

This is also close but still slightly lower.

Let's try: `<calculator> (8890 - 6) - 3 * 1 </calculator>`

`<result> 8881 </result>` This is close but still slightly lower.

Let's try: `<calculator> (8890 - 6) - 3 / 1 </calculator>`

`<result> 8881 </result>` This is close but still slightly lower.

Let's try: `<calculator> ( </calculator> <result> Error: '(' was never closed (<string>, line 1) </result>`

This approach seems to be incorrect.

<reward>:0.0

Prompt+GRPO

<question>:

Using the numbers [3, 8890, 6, 1], create an equation that equals 8888. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<Prompt+GRPO>:

We need to create an equation using the numbers [3, 8890, 6, 1] that equals 8888, using each number only once and employing basic arithmetic operations.

Let's try different combinations:

1. `<calculator> 3 + 8890 + 6 + 1 </calculator> <result> 8900 </result>`
(too high)

2. `<calculator> 3 + 8890 + 6 - 1 </calculator> <result> 8898 </result>`
(too high)

3. `<calculator> 3 + 8890 + 6 * 1 </calculator> <result> 8899 </result>`
(too high)

4. `<calculator> 3 + 8890 - 6 + 1 </calculator> <result> 8888 </result>`
(correct)

So, the equation that equals 8888 is:

`<calculator> 3 + 8890 - 6 + 1 </calculator> <result> 8888 </result>`

The final answer is:#### 3 + 8890 - 6 + 1
<reward>:1.0

<question>:

Using the numbers [27, 72, 511, 9], create an equation that equals -403. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<Prompt+GRPO>:

We need to create an equation using the numbers [27, 72, 511, 9] that equals -403, using each number only once and employing basic arithmetic operations. Let's try different combinations:

1. <calculator>27 + 72 + 511 + 9 </calculator> <result> 619 </result>
(too high)
2. <calculator>27 + 72 + 511 - 9 </calculator> <result> 601 </result>
(too high)
3. <calculator>27 + 72 + 511 * 9 </calculator> <result> 4698 </result>
(too high)
4. <calculator>27 + 72 - 511 + 9 </calculator> <result> -403 </result>
(correct)

So, the equation that equals -403 is:

<calculator>27 + 72 - 511 + 9 </calculator> <result> -403 </result>

The final answer is:#### 27 + 72 - 511 + 9

<reward>:1.0

SFT+GRPO

<question>:

Using the numbers [692, 8, 6, 6], create an equation that equals 66432. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<SFT+GRPO>:

First, let's consider the target number 66432 and the numbers we have: 692, 8, 6, and 6. We need to find a combination of these numbers using basic arithmetic operations to reach 66432. Let's start by exploring multiplication and division since they can significantly change the magnitude of the numbers. First, let's try multiplying 692 by 8:

<calculator> 692 * 8 </calculator> <result> 5536 </result> Next, let's multiply 5536 by 6:

<calculator> 5536 * 6 </calculator> <result> 33216 </result>

Finally, let's multiply 33216 by 2 <result>

(692 * 8) * 6 * 2 = 66432.

<reward>:1.0

<question>:

Using the numbers [54, 749, 1, 9], create an equation that equals -793. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

<SFT+GRPO>:

First, let's try to combine the numbers using basic arithmetic operations to get close to -793.

We have the numbers: 54, 749, 1, 9

Let's start by experimenting with different combinations and operations.

First, let's try subtracting 749 from 54:

<calculator> 54 - 749 </calculator> <result> -695 </result>

Next, let's subtract 9 from -695:

<calculator> 54 - 749 - 9 </calculator> <result> -704 </result>

Finally, let's subtract 1 to get -793:

<calculator> 54 - 749 - 9 - 1 </calculator> <result> -705 </result>

(54 - 749 - 9) - 1

<reward>:0.0