Explore-Execute Chain: Towards an Efficient Structured Reasoning Paradigm

Anonymous ACL submission

Abstract

Many LLMs plan before they act, yet planning and execution are often still entangled in one long generation trace, enforced only through prompts, or split across separate components. We argue that these two stages call for different computation: planning benefits from diversity and breadth, whereas execution demands precision and faithful adherence to a chosen strategy. Treating them as a single undifferentiated chain wastes tokens on routine derivation and makes it costly to explore alternative strategies at test time. We present the **Explore-Execute Chain (E^2C)**, which keeps both stages in one model but separates them structurally: a stochastic *Exploration* phase drafts a concise high-level plan, and a deterministic *Execution* phase carries it out. Causal SFT and RL train this split so that exploration stays informative and execution remains plan-faithful. Once plans are short yet decisive, extra inference compute can be directed to exploration rather than to repeatedly decoding full solutions. On AIME'2024 at K=32, **E^2C-ReAct Loop** reaches 53.3% accuracy with only 12.4k tokens, outperforming Tree-of-Thoughts (N=32: 50.0%, 71.3k). The same structure also supports lightweight domain adaptation: **Exploration-Focused SFT (EF-SFT)** updates only the planning phase, uses 3.5% of the tokens required by standard SFT, and improves medical benchmark accuracy by up to 14.5%. The code and models will be released upon publication.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning, largely propelled by techniques such as Chain-of-Thought (CoT) prompting (Wei et al., 2022). This paradigm has inspired a suite of advanced methods, including sampling multiple reasoning paths for consensus via Self-Consistency (Wang et al., 2022), and exploring the solution space with more complex structures like Tree-of-Thoughts (ToT) (Yao et al., 2023), Graph-of-Thoughts (GoT) (Besta et al., 2023), and Forest-of-Thought (FoT) (Bi et al., 2025). Other approaches focus on iterative refinement through self-correction (Shinn et al., 2023) or problem decomposition (Zhou et al., 2023; Yao et al., 2022).

Despite their success, most of these methods still generate planning and execution within a single undifferentiated reasoning chain. Prior work has explored plan-then-execute patterns through subtask decomposition (Zhou et al., 2023), interleaved reasoning and tool use (Yao et al., 2022), or LLM planners paired with external solvers (Hao et al., 2023; Liu et al., 2023). However, such approaches typically interleave planning with action at similar granularity, rely on prompting rather than training the decomposition, or delegate execution outside the model. They therefore still conflate two fundamentally different computational needs within reasoning: high-level strategic **planning**, which benefits from diversity, and low-level step-by-step **execution**, which requires precision. This entanglement leads to critical inefficiencies. First, the model expends equivalent computational effort on both creative planning and routine calculations (Xu et al., 2025b; Li et al., 2025). Second, greedy decoding restricts the diversity of initial strategies, where a suboptimal early choice can derail the entire reasoning path (Liao et al., 2025; Xu et al., 2025a; Zheng et al., 2025).

Rather than introducing planning, we ask how a single LLM should internalize plan-then-execute reasoning as a trainable, structurally explicit process. We introduce the **Explore–Execute Chain (E^2C)**, which decomposes standard CoT into two distinct output phases within one model (Figure 1, left). The first phase is a highly informative **exploration** stage that produces a concise high-level plan, offering a compact preview of the solution strategy without the cost of full-chain generation. The second phase is a highly deterministic **execu-**

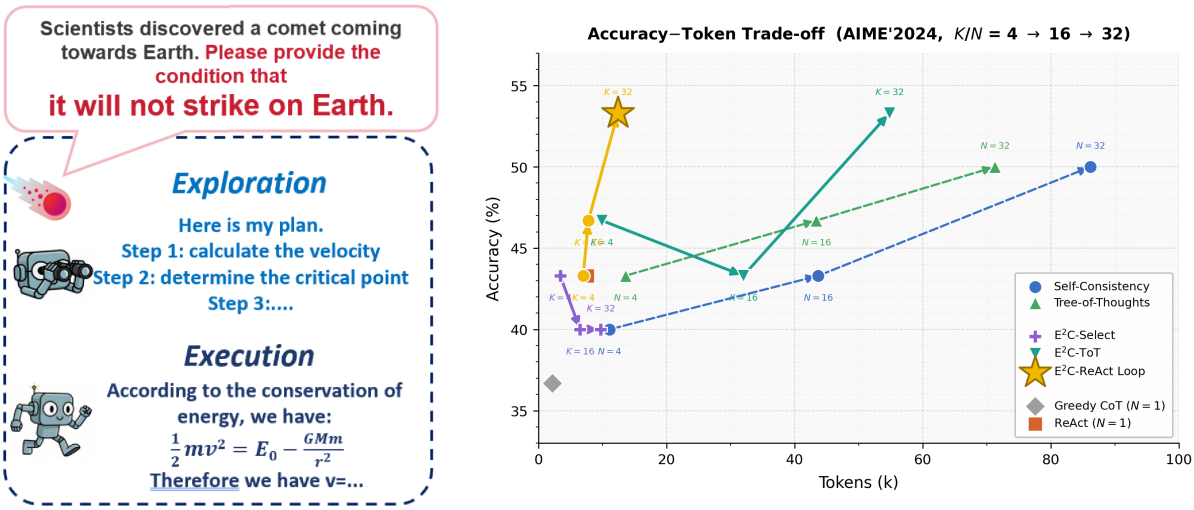

Figure 1: **Left:** A concrete example of **E^{2}C** in action. Given a physics question, the model first produces a concise *Exploration* that outlines the high-level solution strategy (e.g., "Step 1: calculate the velocity; Step 2: determine the critical point..."), then performs a detailed, step-by-step *Execution* (deriving the equations and arriving at the answer). The two stages are structurally separated, keeping planning succinct and computation faithful. **Right:** Accuracy–token trade-off on AIME'2024 at budget K/N=32. E^{2}C-ReAct Loop ($\star$) achieves 53.3% accuracy with only 12.4k tokens, outperforming all full-chain baselines that require 4–7$\times$ more tokens to reach 50.0%.

tion stage that carries out the plan in detailed calculations. Unlike prompt-only decomposition, both phases are produced by the same model and are explicitly trained to satisfy complementary objectives: informative planning and faithful execution.

This decomposition enables a highly efficient **test-time scaling** strategy. Existing methods such as Self-Consistency (Wang et al., 2022), Tree-of-Thoughts (Yao et al., 2023), and ReAct (Yao et al., 2022) all scale full reasoning chains, so their token cost grows in proportion to the budget. E^{2}C breaks this coupling: because Exploration is short and highly informative, additional compute can be concentrated there while Execution remains sparse and deterministic. Each standard TTS strategy has a corresponding E^{2}C variant that operates over plans rather than full chains: multi-sample aggregation becomes plan selection, tree search is restricted to the Exploration stage, and iterative refinement operates over plans rather than complete solutions. As shown in Figure 1 (right), this yields dramatically better accuracy–token efficiency: E^{2}C-ReAct Loop surpasses full-chain baselines while using a fraction of their token budget. We implement this framework using a two-stage (SFT+RL) training pipeline, guided by recent advances in reasoning alignment (Gan et al., 2025; Rafailov et al., 2023).

Our main contributions are summarized as follows:

- We show that, within a single LLM, reasoning can be more effectively organized as two computationally distinct phases: a planning stage that benefits from stochastic diversity and an execution stage that demands precision and determinism. E^{2}C makes this separation explicit in both output structure and training, rather than relying on prompt-level plan-then-execute heuristics alone.
- We introduce a robust two-stage training methodology (SFT+RL) together with a specialized data construction algorithm that ensures the model faithfully adheres to its plans, effectively instilling E^{2}C paradigm and achieving superior performance.
- We demonstrate the efficiency of this framework through two applications: concentrating inference compute on the cheap Exploration stage yields competitive test-time scaling with substantially fewer tokens, while Exploration-Focused SFT (EF-SFT) enables lightweight domain adaptation with significant accuracy gains.

2 Preliminary

We introduce the notation used throughout Methodology. Given a context c (e.g., a question) and a target answer a, the model generates two structured components before producing a. The **Exploration** plan π is a concise high-level strategy. The **Execution** trace $e = (e_1, \ldots, e_T)$ is the detailed step-by-

step derivation that follows π, where T denotes the execution length.

Standard Chain-of-Thought (CoT) couples these two roles in a single autoregressive chain modeled by $p(e \mid c)$. E^{2}C instead factorizes reasoning within one LLM as

$$p'(\pi, e \mid c) = p'(\pi \mid c)\, p'(e \mid \pi, c). \tag{1}$$

The distribution $p'(\pi \mid c)$ controls plan generation and should remain short yet informative. The distribution $p'(e \mid \pi, c)$ controls execution and should follow the chosen plan faithfully. Section 3.1 formalizes these requirements and describes how training enforces them.

3 Methodology

Building on this formulation, we introduce the **Explore-Execute Chain (E^{2}C)** framework. As shown in Fig. 2, we first describe a two-stage training procedure, then present EF-SFT and plan-level test-time scaling.

3.1 Formal Definition of E^{2}C

Building on Eq. equation 1, E^{2}C replaces the coupled process $p(e \mid c)$ with the explore-execute factorization

$$\underbrace{p(e \mid c)}_{\text{Coupled Reasoning Process}} \rightarrow \underbrace{p'(\pi, e \mid c)}_{\text{Explore-Execute Chain}} = \underbrace{p'(\pi \mid c)}_{\text{Highly Informative}} \cdot \underbrace{p'(e \mid \pi, c)}_{\text{Highly Deterministic}} \tag{2}$$

The framework is defined by two core properties:

1. **(Informative Property)**. $p'(\pi \mid c)$ should be highly informative, containing the critical information necessary to solve the problem.
2. **(Deterministic Property)**. $p'(e \mid \pi, c)$ should be highly deterministic, meaning it must fully leverage the informative π.

Naturally, we semantically design π to represent high-level strategies, while e entails detailed calculations that follow π.

3.2 2-Stage Training Procedure: SFT and RL

We introduce a two-stage training procedure to achieve the proposed Prop. 1 and Prop. 2. Stage 1 is Supervised Fine-Tuning (SFT), in which we construct a synthetic dataset and perform SFT to achieve a paradigm shift in reasoning and satisfy the informative Prop. 1. We do not rely solely on prompting to accomplish this paradigm transition because prompting is unstable and leads to a more significant performance drop compared to SFT training. Detailed results are presented in Table 1. Stage 2 employs Reinforcement Learning (RL), which incorporates a λ-coefficient on the advantage to appropriately leverage Prop. 1, thereby accelerating convergence and enhancing the determinism of execution to satisfy Prop. 2.

3.2.1 Stage 1: E^{2}C Supervised Finetuning

To support structured reasoning, we construct a dedicated SFT dataset through synthetic generation. A naive method is to first sample an execution trace from the base model and then summarize it into an exploration step. However, this approach is flawed: the execution is generated from $p(e \mid c)$ rather than the desired $p'(e \mid \pi, c)$, effectively hacking the causal structure. As a result, the model learns to ignore the exploration and directly mimic the base model's execution distribution, violating the intended information bottleneck.

Our method, formalized in Algorithm 1, reverses this order: for each question, we first generate a full solution, distill it into an exploration step, and then prompt the model to produce a new execution conditioned on that plan. This enforces a causal dependency from exploration to execution, which is crucial for Prop. 2. The solution can also come from the ground truth. To enable a fair comparison and minimize dataset selection constraints while avoiding the introduction of extra variables, we specifically use samples from the Base LLM in our comparison experiments

3.2.2 Stage 2: E^{2}C Reinforcement Learning

To emphasize informative reasoning, we extend hierarchical weighting (Wang et al., 2025b) by assigning a higher coefficient λ to exploration tokens, which accelerates convergence (Prop. 1), while the entropy-reduction effect of reinforcement learning supports determinism (Prop. 2). The training objective is defined as

$$\mathcal{L}_{\text{clip}} = \frac{1}{G} \sum_{i,t} \min\Big(r_{i,t}\, \lambda_{i,t}\, \hat{A}_{i,t},\ \text{clip}(r_{i,t},\, 1-\varepsilon,\, 1+\varepsilon)\, \lambda_{i,t}\, \hat{A}_{i,t} \Big). \tag{3}$$

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}[\mathcal{L}_{\text{clip}}] - \beta\, D_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big]. \tag{4}$$

where $\hat{A}_{i,t} = (r_{i,t} - \bar{r}_i)/\sigma_i$ and $r_{i,t} = r_{\text{answer}} + r_{\text{format}}$. The reward r_{answer} measures answer correctness, while r_{format} consists of a length reward (r_{length}) designed to prevent overly long and repetitive answers and an instruction reward (r_{instr}), quantifies the alignment between exploration and execution, ensuring that exploration trajectories

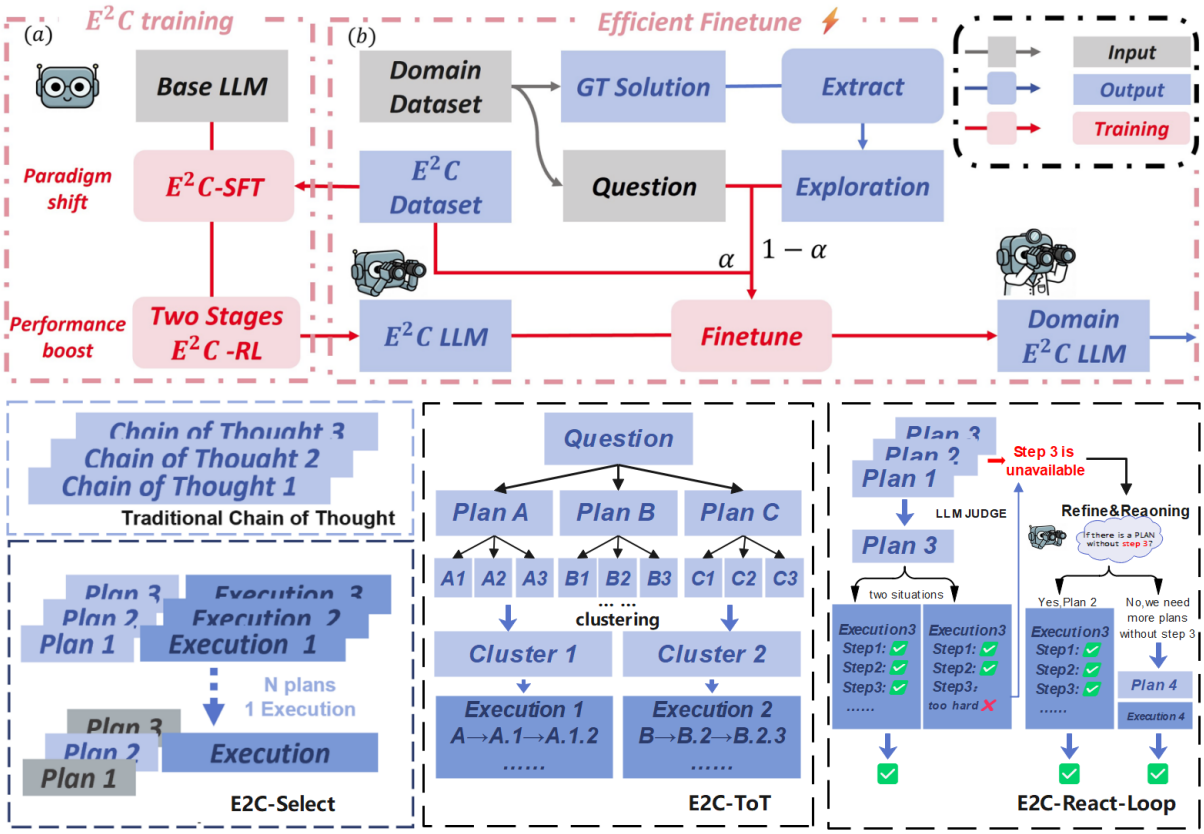

Figure 2: **Overview of E²C method.** The approach begins with E²C-SFT to achieve a paradigm shift, followed by a two-stage E²C-RL process that leverages the decomposition advantage of the new paradigm to boost performance. The resulting E²C-LLM can be efficiently adapted to new domains via EF-SFT. At test time, the high informativeness and compact length of the Exploration stage allows additional compute to be concentrated on plan-level search, selection, and refinement rather than on costly full-chain generation.

approximate optimal execution strategies.The detailed description for r_{format} can be found in Appendix A.2.1.

We adopt a two-stage training procedure. In the first stage, a higher temperature τ_1 and larger rollout number k_1 are used for one epoch, encouraging broad exploration of the action space and fostering self-correction to mitigate the overly rigid adherence to the exploration plan that results. In the second stage, we reduce the temperature to τ_2 and the rollout number to k_2, again for one epoch, and assign the advantage coefficient $\lambda_{i,t} = \lambda_{exp} > 1$ for the exploration tokens in the GRPO update. This modification explicitly prioritizes high-level reasoning in the policy gradient, thereby achieving faster and more stable convergence.

The behavior of the trained agent can be formalized by analyzing the modified GRPO objective in Eq. (2). We highlight the following quantified properties:

1. Update emphasis. Assigning $\lambda_{i,t} = \lambda_{\text{exp}} > 1$ for $t \in T_{\text{exp}}$ and $\lambda_{i,t} = 1$ for $t \in T_{\text{exe}}$ yields

$$\frac{\mathbb{E}\big[\|g_{i,t}\|^2 \mid t \in T_{\text{exp}}\big]}{\mathbb{E}[\|g_{i,t}\|^2 \mid t \in T_{\text{exe}}]} \gtrsim \lambda_{\text{exp}}^2, \tag{5}$$

so exploration tokens receive disproportionately larger gradient updates, strengthening the planning phase. Entropy dynamics in Appendix A.5 confirm the effect of λ_{exp}.

2. Deterministic execution. Stage-2 RL (lower temperature, fewer rollouts) drives

$$\mathbb{E}_{t\in T_{\text{exe}}}[\Delta_{i,t}] \nearrow, \qquad \mathbb{E}_{t\in T_{\text{exe}}}\big[H(\pi_\theta(\cdot \mid q, o_{<t}))\big] \searrow, \tag{6}$$

yielding faithful, low-variance execution.

3. Plan sensitivity. The expected update sign for exploration tokens satisfies

$$\mathbb{E}\big[\text{sgn}(g_{i,t}) \mid t \in T_{\text{exp}}\big] \propto \text{sgn}\Big(\mathbb{E}[\hat{A}_{i,t}^{\text{plan}}]\Big), \tag{7}$$

so high-quality plans are amplified while poor plans are suppressed.

3.3 Efficient Adaptation and Inference with E^{2}C

The modularity of our E^{2}C framework enables efficient strategies for both domain adaptation at training time and scaled aggregation at test time.

Exploration-Focused SFT (EF-SFT). For domain adaptation, we introduce EF-SFT. This method leverages the transferable nature of the execution component by exclusively fine-tuning on the exploration segments from domain-specific examples. These segments are mixed with the base E^{2}C dataset at a controlled ratio α, allowing the model to efficiently learn new reasoning strategies while maintaining its core capabilities. This targeted approach significantly reduces the data and computational requirements for adaptation. A detailed algorithm can be found in the Appendix 2.

E^{2}C Test-Time Scaling: Scale Exploration, Execute Selectively. Standard TTS methods scale full reasoning chains: Self-Consistency samples N complete solutions, Tree-of-Thoughts builds a search tree over intermediate reasoning steps, and ReAct iteratively reissues full solution attempts. Because Execution dominates the token budget of each chain, the cost of these methods grows proportionally with N.

E^{2}C breaks this proportionality. Since the Exploration stage is short ($\leq$512 tokens per plan) yet highly informative, we can afford to generate or revise many plans at a small fraction of the cost of one full Execution. This gives each standard TTS strategy a more efficient E^{2}C counterpart:

- **E^{2}C-Select (Plan Selection):** the plan-level counterpart of multi-sample aggregation. Sample K exploration plans, use the model as a self LM-judge to select the single best plan, and execute it once.
- **E^{2}C-ToT (Exploration-Only Tree Search):** the plan-level counterpart of tree search over full chains. Build a ToT-style tree solely over the Exploration stage, cluster the resulting plan leaves, and execute only representative ones.
- **E^{2}C-ReAct Loop (Execute-Refine Loop):** the plan-level counterpart of iterative full-chain refinement. Execute a selected plan; if execution stalls or fails to produce a valid answer, revise or replace the plan using the failure as feedback, then retry. All iterative search is confined to the compact plan space.

The three variants share a common explore-select-execute-refine template; Algorithm 3 in the Appendix formalizes this procedure, with each variant instantiating a different selection or refinement operator.

4 Experiments

4.1 Training Protocols

We adapt our training codebase from verl (Sheng et al., 2024). The **E^{2}C-SFT** model was trained for one epoch on a 50k-sample synthetic dataset built from Openr1-math (deepseek, 2025) with the causal pipeline in Algorithm 1, then further refined with our two-stage **E^{2}C-RL** on DAPO-17K (Yu et al., 2025). A standard **GRPO** baseline was trained for five epochs on the same data. For domain adaptation, **EF-SFT** uses a targeted 50k-sample exploration-only subset mixed with 10% regularization data, compared against standard SFT trained on the full ReasonMed dataset.

4.2 Experiments Setup

Mathematical reasoning. *Models:* Qwen3-8B and Qwen3-4B in non-thinking mode, comparing E^{2}C-(SFT+RL) with matched GRPO baselines (Sec. 4.1). *Benchmarks:* AIME'24, AIME'25, MATH500, Algebra, Minerva, AMC23, and Olympiad Bench (He et al., 2024). *Evaluation:* Pass@1 accuracy averaged over 8 independent samples.

Medical reasoning. *Models:* Qwen3-8B with math-trained E^{2}C-(SFT+RL) for zero-shot transfer, and Qwen3-8B / Llama3.1-8B for domain adaptation. *Benchmarks:* MedQA (Jin et al., 2021), MedMCQA (Pal et al., 2022), and six MMLU (Hendrycks et al., 2020) medical subsets. *Adaptation setup:* EF-SFT is trained on a 50k-sample exploration-only subset from ReasonMed with 10% regularization data (10M tokens), compared against standard SFT on the full ReasonMed corpus (286M tokens).

Test-time scaling. *Model:* a single checkpoint, Qwen3-8B+E^{2}C-(SFT+RL). *Benchmark:* AIME'2024. *Protocol:* E^{2}C-Select, E^{2}C-ToT, and E^{2}C-ReAct Loop are compared with Self-Consistency, ToT, and ReAct at budgets K or $N \in \{4, 8, 16, 32\}$. Sampled decoding uses temperature 0.9. We report Pass@1 accuracy and average tokens per question; full method details are in Appendix A.4.

Table 1: Performance comparison of Qwen3 models (non-thinking mode) on mathematical reasoning benchmarks. All results are reported as Pass@1 accuracy, with an 8-sample average.

Model	AIME'24	AIME'25	MATH500	Algebra	Minerva	AMC23	Olympiad	Avg Acc	Avg Length
Qwen3 8B Series									
Qwen3-8B+GRPO (Baseline)	36.9	**34.4**	**88.2**	88.2	33.1	79.3	60.0	60.0	1429.46
Qwen3-8B+E^{2}C-SFT+GRPO	37.5	32.5	83.5	86.6	30.8	76.3	56.8	57.7	1309.62
Qwen3-8B+E^{2}C-(SFT+RL)	**40.6**	33.8	87.7	**90.9**	**35.8**	**80.3**	**61.3**	**61.5**	1476.41
Qwen3 4B Series									
Qwen3-4B+GRPO (Baseline)	28.8	**30.6**	84.6	84.4	33.5	75.8	57.8	56.5	1263.15
Qwen3-4B+E^{2}C-SFT+GRPO	28.8	26.9	85.9	83.3	33.2	75.7	55.3	55.2	1324.18
Qwen3-4B+E^{2}C-(SFT+RL)	**37.5**	30.0	**86.1**	**84.8**	**34.0**	**78.3**	**58.4**	**58.4**	1456.34
Ablation Studies									
Qwen3-8B+Prompt (Zero-shot)	21.9	18.8	76.3	80.5	30.9	50.7	45.8	46.6	1142.38
Qwen3-8B+E^{2}C-SFT	23.1	21.9	75.8	80.5	31.5	51.5	43.2	46.8	1162.89

4.3 Results

Mathematical Reasoning Benchmark Results We conduct a sanity check comparing our E^{2}C models (Qwen3-8B/4B+E^{2}C-(SFT+RL)) against GRPO baselines, as shown in Table 1. Our approach outperforms the baselines by 1.5% (8B) and 1.9% (4B), validating the effectiveness of the decomposition strategy. Notably, while paradigm shifts typically risk performance degradation, our method successfully maintains and enhances model capability through careful training design. The full E^{2}C framework ultimately surpasses the GRPO baseline by leveraging the decomposed structure, establishing a solid foundation for efficient test-time scaling. Training-stage ablations (Table 1, bottom rows) are analyzed in Section 4.4.

Medical Reasoning Benchmark Results Table 2 presents the medical reasoning performance across three experimental settings. First, we establish competitive baselines by comparing against leading domain-specific 7B-8B models (HuatuoGPT-o1-7B (Wang et al., 2024), ReasonMed-7B (Sun et al., 2025)) and an open-source 14B medical LLM (Baichuan-M1-14B (Wang et al., 2025a)), with Qwen3-8B (Yang et al., 2025) serving as our base model reference.

For domain adaptation, we evaluate our EF-SFT approach (Section 3.3) against standard SFT on both Llama3.1-8B (Dubey et al., 2024) and Qwen3-8B architectures. As shown in Table 2, EF-SFT achieves significant improvements of 3.9% (Qwen3-8B) and 14.5% (Llama3.1-8B) over standard SFT, while using only 3.5% of the training tokens. The zero-shot transfer results further demonstrate that our mathematically-trained RL models attain performance comparable to specialized medical LLMs, validating the strong cross-domain generalization capability of our method.

Test-Time Scaling Performance and Efficiency Analysis Table 3 shows a consistent pattern: for each full-chain baseline, the corresponding E^{2}C variant achieves higher accuracy at lower token cost by scaling only the Exploration stage. The strongest result is **E^{2}C-ReAct Loop** at K=32, which reaches **53.3%** with only 12.4k tokens, outperforming Self-Consistency at N=32 (50.0%, 86.2k tokens) and Tree-of-Thoughts (50.0%, 71.3k tokens).

4.4 Ablations and Analysis

We ablate the main design choices behind E^{2}C across training, data construction, and domain adaptation (Table 4; bottom rows of Table 1).

Training pipeline. On Qwen3-8B, prompt-only E^{2}C formatting reaches 46.6% average accuracy, nearly matching zero-shot performance and showing that prompting alone cannot reliably instill the explore-execute interface. E^{2}C-SFT raises this to 46.8%, confirming that explicit SFT is needed for the paradigm shift. Replacing generic GRPO with our E^{2}C-RL further improves 8B average accuracy from 57.7% to 61.5% (+3.8%), with a similar +3.2% gain on 4B, indicating that RL benefits come from exploiting decomposition rather than additional data alone.

Causal data construction (Part A). As discussed in Section 3.2.1, reverse-causal summarization lets the model learn to bypass exploration. Table 4 quantifies this with *Plan-Guided* adherence (execution follows its plan) and *Plan-Free* adherence (execution answers without the plan). The flawed pipeline scores 0.499 and 0.864, respectively, whereas the causal pipeline in Algorithm 1 reaches 0.998 and 1.000, enforcing the intended

Table 2: Performance Comparison of Models with Different Training Processes: Our inference paradigm demonstrates superior generalization, while EF-SFT shows improved efficiency and robustness. The six columns from Anatomy (AN), Clinical Knowledge (CK), College Biology (CB), College Medicine (CM), Medical Genetics (MG), and Professional Medicine (PM) are validation subsets of the MMLU benchmark.

Model	MedQA	MedMCQA	AN	CK	CB	CM	MG	PM	#Med-Tokens	Avg
External Baselines										
HuatuoGPT-o1-7B	68.4	57.5	71.9	78.5	88.2	67.6	80.0	77.6	-	73.7
Baichuan-M1-14B	76.5	65.2	77.3	83.6	87.9	80.7	89.1	88.8	-	81.1
ReasonMed-7B	66.9	65.1	75.6	79.3	79.2	73.4	85.0	80.9	-	75.7
Our Method and Ablations										
Qwen3-8B	71.4	59.5	68.0	81.6	87.5	78.0	85.0	83.8	-	76.8
Qwen3-8B + GRPO	74.0	60.6	75.0	80.9	91.3	81.8	90.4	**86.2**	-	79.1
Qwen3-8B+E^{2}C-(SFT+RL)	**74.5**	**63.1**	**77.0**	**82.2**	**92.0**	**83.0**	**92.8**	86.0	-	**81.1**
SFT Models (using medical data)										
Qwen3-8B + standard SFT	58.2	52.3	68.8	80.8	89.0	73.7	83.3	79.0	286M	73.1
Qwen3-8B + E^{2}C-SFT + EF-SFT	**65.8**	**58.2**	**72.3**	**83.8**	**89.2**	**79.7**	**87.6**	**86.2**	10M	**77.1**
Llama3.1-8B + ReasonMed SFT	42.0	36.8	45.9	55.4	61.8	43.2	38.1	56.9	286M	47.5
Llama3.1-8B + E^{2}C-SFT + EF-SFT	**60.3**	**53.2**	**61.8**	**69.8**	**75.9**	**64.9**	**82.0**	**72.5**	10M	**67.5**

Table 3: Test-Time Scaling Performance on AIME'2024 Benchmark (Qwen3-8B+E2C(SFT+RL)). Methods are grouped by scaling strategy: for each full-chain baseline, the corresponding E^{2}C variant (which concentrates additional compute on the short Exploration stage) is shown directly below. E^{2}C variants consistently achieve higher accuracy with substantially fewer tokens.

Method	Budget Level 1		Budget Level 2		Budget Level 3		Budget Level 4	
	Acc. (%)	Tokens (k)	Acc. (%)	Tokens (k)	Acc. (%)	Tokens (k)	Acc. (%)	Tokens (k)
Greedy CoT ($N = 1$)	36.7	2.2	*(single decoding pass; N/A at higher budgets)*					
Multi-sample aggregation								
Self-Consistency	40.0 (N=4)	11.1	40.0 (N=8)	21.8	43.3 (N=16)	43.7	50.0 (N=32)	86.2
E^{2}C-Select (Self LM-Judge)	43.3 (K=4)	3.4	26.7 (K=8)	4.6	40.0 (K=16)	6.5	40.0 (K=32)	9.7
Tree search								
Tree-of-Thoughts (ToT)	43.3 (N=4)	13.6	43.3 (N=8)	27.1	46.7 (N=16)	43.4	50.0 (N=32)	71.3
E^{2}C-ToT	**46.7 (K=4)**	**9.9**	**46.7 (K=8)**	**18.3**	43.3 (K=16)	**32.0**	**53.3 (K=32)**	**54.8**
Iterative refinement								
ReAct ($N = 1$)	43.3	7.6	*(single-loop reference; N/A at higher budgets)*					
E^{2}C-ReAct Loop	43.3 (K=4)	**7.0**	43.3 (K=8)	**5.7**	**46.7 (K=16)**	**7.8**	**53.3 (K=32)**	**12.4**

$p'(e \mid \pi, c)$ dependency required by Prop. 2.

EF-SFT training budget (Part B). Because EF-SFT updates only exploration segments, we ask how much domain-specific training is enough. On medical targets, 300 iterations (≈5k samples) yields the best Anatomy (72.3%) and Clinical Knowledge (83.8%) scores. Extending to 2000 iterations improves in-domain GSM8K (84.0) but hurts MedQA (63.5 vs. 66.4 at 100 iterations), suggesting overfitting to exploration phrasing without improving transferable execution. We therefore use 300 iterations in all EF-SFT experiments.

Regularization mixing (Part C). Mixing 10% base E^{2}C-SFT data during EF-SFT preserves math ability (GSM8K 83.3, MATH 71.2) while adapting medical reasoning. Removing regularization (α=0%) drops MedQA to 63.5, indicating that execution skills erode without replay data. Using the full exploration-execution sequence (α=100%) also underperforms, because it dilutes the exploration-only adaptation signal. Interestingly, math regularization at α=10% outperforms medically matched regularization on average, so target-domain regularization data is not required for stable EF-SFT.

5 Limitations and Future Works

Error propagation through deterministic execution. Because the execution stage is designed to faithfully follow its conditioning plan, a flawed plan is not self-corrected during execution but instead carried out in full. This asymmetry means the framework's overall accuracy is disproportionately

Table 4: Ablation studies: (A) data construction strategy, (B) EF-SFT training steps, (C) regularization data mixing.

Part A: SFT Data Construction Strategy

SFT Data Strategy	Plan-Guided	Plan-Free
Flawed (Reverse-Causal Summary)	0.499	0.864
Proposed (Causal Generation)	**0.998**	**1.000**

Part B: EF-SFT Training Steps

Steps	GSM8K	Anatomy	CK	MedQA	MATH
SFT + 100 iter	83.0	70.4	80.6	**66.4**	70.5
SFT + 300 iter	83.3	**72.3**	**83.8**	66.1	**71.2**
SFT + 2000 iter	**84.0**	69.8	78.4	63.5	71.1

Part C: Regularization Data Mixing

Setting	GSM8K	Anatomy	CK	MedQA	MATH
Medical Reg. (α=10%)	82.1	**74.4**	79.5	66.8	69.5
Math Reg. (α=10%)	**83.3**	72.3	**83.8**	66.1	**71.2**
Medical Reg. (α=100%)	80.1	70.0	81.5	**67.0**	65.8
No Reg. (α=0%)	82.1	68.2	77.8	63.5	70.8

sensitive to plan-level errors, with no internal recovery mechanism within a single explore-execute pass.

Future Work. The explore-then-execute decomposition of E^{2}C maps naturally onto agentic settings. A direct extension is to treat the Exploration stage as a lightweight action-planning module that produces a structured sequence of tool calls or sub-tasks, while the Execution stage carries out each action step. This allows the plan-level scaling strategies developed here (selection, tree search, and iterative refinement) to be applied directly to multi-step tool-using agents.

6 Related Work

From Chain-of-Thought to Structured Reasoning: Chain-of-Thought (CoT) prompting (Wei et al., 2022) significantly improves LLM reasoning, but its linear nature has motivated more robust structured paradigms that explore diverse reasoning paths (Chen et al., 2025). These include parallel sampling methods such as Self-Consistency (Wang et al., 2022; Wan et al., 2025), and more complex search structures including trees (ToT) (Yao et al., 2023), graphs (GoT) (Besta et al., 2023; Yao et al., 2024), and forests (FoT) (Bi et al., 2025). While these paradigms expand the search space, they typically do not enforce a compact, trainable plan-to-execution interface within a single model. E^{2}C targets this gap through explicit phase separation and joint training.

Planning and Decomposition in LLM Reasoning: Planning before execution is already well studied in LLM reasoning, from subtask decomposition (Zhou et al., 2023; Press et al., 2022) and interleaved tool use (Yao et al., 2022; Schick et al., 2023; Patil et al., 2023) to LLM planners paired with external solvers (Hao et al., 2023; Liu et al., 2023) or multi-agent systems (Yuan et al., 2024). E^{2}C differs in scope and mechanism: it keeps both planning and execution inside one model, but separates them into structurally distinct phases with a compact plan interface, and trains that factorization directly via SFT and RL rather than treating it as a prompting convention. This design enables stable plan-level test-time scaling and exploration-only domain adaptation.

Test-Time Scaling and Reasoning Efficiency: Test-time scaling (TTS) improves performance by increasing inference-time compute (Snell et al., 2024; Wu et al., 2025). However, methods like Self-Consistency (Wang et al., 2022) and Tree-of-Thoughts (Yao et al., 2023) generate multiple full-length reasoning chains, making their token cost grow proportionally with the budget. E^{2}C contributes a structurally motivated TTS strategy: by decoupling Exploration from Execution, it restricts search, selection, and refinement to the compact planning stage, so the token cost of scaling is determined by plan length rather than full solution length.

7 Conclusion

Through the proposed Explore-Execute Chain (E^{2}C), we present a reasoning framework that makes plan-then-execute structure explicit within a single LLM, enhancing efficiency and interpretability relative to undifferentiated CoT and prompt-only decomposition. The framework concentrates information in exploration, enabling domain adaptation with only 3.5% of training tokens and a strong performance-cost trade-off on complex reasoning benchmarks. By restricting test-time scaling to compact plans, E^{2}C further offers a practical path toward efficient inference at higher compute budgets. We hope this decomposition encourages further work on trainable reasoning structure, where exploration and execution can be optimized, audited, and scaled independently within a unified model. Our results across mathematical and medical benchmarks, together with ablations on data construction and plan-level scaling, support explicit explore-execute factorization as a practical direction for both adaptation and inference efficiency.

Ethics Statement

This work studies a reasoning framework, the **Explore–Execute Chain (E^2C)**, which separates lightweight exploratory sketches from a final execution step to improve efficiency, transparency, and controllability of LLM reasoning. Our experiments fine-tune and evaluate general-purpose LLMs on publicly available benchmarks (e.g., mathematics and domain reasoning datasets). We do not collect new human data, do not involve human or animal subjects, and do not process personally identifiable or sensitive information. Any third-party datasets used in this paper are publicly released for research purposes by their respective providers; we follow their licenses and usage terms. E^2C increases interpretability by exposing intermediate "exploration" traces, which can facilitate auditing and discourage over-reliance on hidden chain-of-thought. This study complies with the conference Code of Ethics.

Reproducibility Statement

We have made extensive efforts to ensure the reproducibility of our work. All code used in this paper will be publicly released to facilitate independent verification and further research. We describe our experimental setup in Sec. 4.1. Detailed hyperparameters for training E^2C-SFT, E^2C-RL, GRPO, and EF-SFT are provided in Appendix A.2.1. Detailed setup for TTS experiments can be found in Appendix A.4. We also include the prompt templates for data generation, the zero-shot prompt model, plan selection, execution, and exploration refinement in Appendix A.6.

Use of Large Language Models

We utilized a large language model to enhance the language and clarity of our manuscript. Specifically, we employed Gemini 2.5 Flash with the following prompt to refine the initial draft: *I am writing an academic paper in English. Please polish the following draft so that it adheres to the conventions of academic writing.*

References

Anja Achtziger and Peter M Gollwitzer. 2007. Rubicon model of action phases.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Kedar Tatwawadi, Joana Einsiedler, Daria Costanzo, Gregor J. Räbsamen, Michael Wand, Hermann sundry, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *Preprint*, arXiv:2308.09687.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *Preprint*, arXiv:2412.09078.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning in large language models. *Preprint*, arXiv:2503.09567.

deepseek. 2025. Open r1: A fully open reproduction of deepseek-r1. `https://github.com/huggingface/open-r1`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Zeyu Gan, Hao Yi, and Yong Liu. 2025. CoT-Space: A theoretical framework for internal slow-thinking via reinforcement learning. *Preprint*, arXiv:2509.04027.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward S. Hu, Kwangjun Ahn, Qinghua Liu, Haoran Xu, Manan Tomar, Ada Langford, Dinesh Jayaraman, Alex Lamb, and John Langford. 2025. The belief state transformer. In *The Thirteenth International Conference on Learning Representations*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2025. Compressing chain-of-thought in llms via step entropy. *Preprint*, arXiv:2508.03346.

Baohao Liao, Hanze Dong, Yuhui Xu, Doyen Sahoo, Christof Monz, Junnan Li, and Caiming Xiong. 2025. Fractured chain-of-thought reasoning. *Preprint*, arXiv:2505.12992.

Boi-Faltings Liu, Zhang-Wei Liu, Ruibo Jiang, Yisong Lyu, Yizhou Du, F. Wu, and Yu-Feng Liu. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Ofir Press, Or Yoran, Timo Schick, Idan Schmid, Ayal Fisch, Yoav Goldberg, and Kanishka Misra. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Tsvigun, Gautier Cances, and Najma Smaili. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. 2025. Spurious rewards: Rethinking training signals in rlvr. *Preprint*, arXiv:2506.10947.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Yu Sun, Xingyu Qian, Weiwen Xu, Hao Zhang, Chenghao Xiao, Long Li, Yu Rong, Wenbing Huang, Qifeng Bai, and Tingyang Xu. 2025. Reasonmed: A 370k multi-agent generated dataset for advancing medical reasoning. *arXiv preprint arXiv:2506.09513*.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

Bingning Wang, Haizhou Zhao, Huozhi Zhou, Liang Song, Mingyu Xu, Wei Cheng, Xiangrong Zeng, Yupeng Zhang, Yuqi Huo, Zecheng Wang, Zhengyun Zhao, and 1 others. 2025a. Baichuan-m1: Pushing the medical capability of large language models. *arXiv preprint arXiv:2502.12671*.

Haozhe Wang, Qixin Xu, Che Liu, Junhong Wu, Fangzhen Lin, and Wenhu Chen. 2025b. Emergent hierarchical reasoning in llms through reinforcement learning. *arXiv preprint arXiv:2509.03646*.

Junying Wang, Zhaonan Li, Renfeng Pu, Saijiang Shi, Yitong Meng, Zhaokun Wang, Yixin Liu, Jianing Zhou, Wenjia Zhang, Jialiang Chen, Yefeng Zheng, and Hong-Yin Mey. 2024. HuatuoGPT, a general-purpose chinese medical large language model. *Preprint*, arXiv:2405.18524.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025a. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *Preprint*, arXiv:2505.11484.

Zenan Xu, Zexuan Qiu, Guanhua Huang, Kun Li, Siheng Li, Chenchen Zhang, Kejiao Li, Qi Yi, Yuhao Jiang, Bo Zhou, Fengzong Lian, and Zhanhui Kang. 2025b. Adaptive termination for multi-round parallel reasoning: An universal semantic entropy-guided framework. *Preprint*, arXiv:2507.06012.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and et al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Sha, Silvio Savarese, and Tao an. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Yao Yao, Zuchao Li, and Hai Zhao. 2024. GoT: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2901–2921, Mexico City, Mexico. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Siyuan Yuan, Kairui Song, Jia-Hao Chen, Xiao-Hui Tan, Dian-Hui Li, and Dong-Sheng Yang. 2024. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *Preprint*, arXiv:2405.16510.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *Preprint*, arXiv:2406.07394.

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, Qian Liu, Ge Zhang, and Zejun Ma. 2025. First return, entropy-eliciting explore. *Preprint*, arXiv:2507.07017.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*.

A Appendix

A.1 Cognitive Model Analysis

In this section, we analyze cognitive models to derive high-level design insights for our method.

A.1.1 Rubicon Model of Action Phases

The **Rubicon Model of Action Phases** (Achtziger and Gollwitzer, 2007), proposed by **Heckhausen** and **Gollwitzer**, provides a framework for how individuals prepare for and pursue goals. It divides goal pursuit into four stages: **goal setting**, **planning**, **action**, and **evaluation**.

1. **Goal Setting**: Individuals identify and adopt a goal, motivated by a need or desire.

2. **Planning**: After a goal is adopted, individuals generate strategies to achieve it and assess their potential effectiveness.

3. **Action**: Once a strategy is selected, the individual commits to it and executes it. Crossing the "Rubicon" marks this commitment and the transition to action.

4. **Evaluation**: Outcomes are assessed to inform adjustments to subsequent plans or actions.

A key contribution of the Rubicon Model is the sharp distinction between planning and execution. After commitment (crossing the Rubicon), attention is devoted to execution rather than continued exploration or second-guessing. This separation mitigates cognitive overload that could arise from ongoing re-evaluation during task execution.

A.1.2 Connecting E^{2}C with the Method

We formally express E^{2}C as

$$\underbrace{p(e \mid c)}_{\text{Coupled Reasoning Process}} \rightarrow \underbrace{p'(\pi, e \mid c)}_{\text{Explore-Execute Chain}} = \underbrace{p'(\pi \mid c)}_{\text{Highly Informative}} \cdot \underbrace{p'(e \mid \pi, c)}_{\text{Highly Deterministic}} \tag{8}$$

$p'(\pi \mid c)$ **as the Planning Phase**: In the Rubicon framework, planning entails generating candidate strategies. Analogously, in E^{2}C, $p'(\pi \mid c)$ produces multiple candidate plans π from context c. These plans are highly informative, capturing the critical information needed to solve the task. This exploration corresponds to the goal-setting and planning stages, where alternatives are considered before selection.

$p'(e \mid \pi, c)$ **as the Execution Phase**: Once plans are available, E^{2}C transitions to execution. The

distribution $p'(e \mid \pi, c)$ reflects a highly deterministic process that follows the selected plan π under context c. This mirrors the action phase of the Rubicon Model: the agent executes the committed plan without revisiting discarded alternatives.

Thus, the separation between $p'(\pi \mid c)$ and $p'(e \mid \pi, c)$ in E^{2}C parallels the explore–then–execute dynamics of the Rubicon Model: first enumerate options, then execute deterministically.

A.1.3 Cognitive and Computational Efficiency

Separating exploration from execution confers efficiency benefits in both cognition and computation. Cognitively, once commitment occurs, resources are focused on carrying out the chosen plan without distraction from alternatives. Computationally, E^{2}C avoids the overhead of re-evaluating multiple plans during execution. The deterministic execution phase concentrates compute on following the selected plan, yielding faster and more reliable performance than continually interleaving exploration with action.

A.1.4 Interpretability and Transparency

The exploration–execution split also improves **interpretability**. In the Rubicon Model, one can explain an action by the plan selected during the planning stage. Likewise, E^{2}C makes the reasoning path explicit: multiple candidate plans are generated (exploration), and one is chosen and followed (execution). This transparency further supports **scalability**: the exploration component can be adapted to new tasks and domains, while the execution component remains stable, enabling flexible and extensible reasoning across settings.

A.2 The Details of The Experiments

In this section, we introduce the details of our main experiments in the main paper for reproducibility purposes, including the detailed hyperparameter settings and the reward designs.

A.2.1 Hyperparameter Settings

E^{2}C-SFT and EF-SFT Training For both E^{2}C-SFT and EF-SFT training, the hyperparameters are summarized in Table 5:

E^{2}C-RL and GRPO Training The hyperparameters for E^{2}C-RL and GRPO training are summarized in Table 6, where the experiments include E^{2}C Stage 1 (E2C-stg1), E^{2}C Stage 2 (E2C-stg2), and GRPO:

A.2.2 Reward Details for RL training

Format Reward Calculation for E^{2}C Training For the **E^{2}C** training, the format reward consists of two components: the length reward and the instruction reward. These rewards are computed as follows:

Length Reward: This reward measures how well the output length matches the expected length. It is computed as:

$$r_l = -\text{clip}\left(0, 1, \frac{L - L_{valid}}{L_{buffer}}\right)$$

where: L is the length of the generated output; L_{valid} is the length of the valid portion of the response; L_{buffer} is the overlong buffer length.

Instruction Reward: The instruction reward is specific to the **E^{2}C** model and is added to the reward function when it comes to E^{2}C model. This reward measures the alignment between the instructions generated during the exploration phase and the execution phase. It is computed by extracting the step titles from both the exploration and execution phases using regular expressions. Denote these sets of instructions as S_1 (exploration) and S_2 (execution). The instruction reward is defined as:

$$r_{\text{instr}} = 0.1 * (\frac{|S_1 \cap S_2|}{\max(|S_1|, |S_2|)} - 1)$$

where: S_1 is the set of instructions generated during the exploration phase; S_2 is the set of instructions generated during the execution phase; $|S_1 \cap S_2|$ is the intersection of the sets S_1 and S_2; $\max(|S_1|, |S_2|)$ is the maximum size of the two sets.

The instruction reward incentivizes the model to generate instructions that align well between the exploration and execution phases, encouraging consistency. This reward is crucial for **E^{2}C** models to ensure that the reasoning process is coherent between the exploration and execution stages.

Format Reward Calculation for GRPO Training For **GRPO** training, the format reward is simpler and consists solely of the length reward, which is computed using the same formula as in E^{2}C:

$$r_l = -\text{clip}\left(0, 1, \frac{\text{length}_{\text{output}} - \text{valid}_{\text{length}}}{\text{buffer}_{\text{length}}}\right)$$

In GRPO, no instruction reward is applied, and the focus is entirely on the length of the response,

Hyperparameter	Value
Learning Rate	1.0×10^{-5}
Optimizer	Adam(Kingma, 2014) ($\beta_1 = 0.9, \beta_2 = 0.95$)
Weight Decay	0.01
Learning Rate Scheduler	Cosine with 10% warmup ratio
Batch Size	160
Micro-batch Size per GPU	20
Gradient Clipping	1.0
Total Epochs	1

Table 5: Hyperparameters for E^{2}C-SFT and EF-SFT Training

Hyperparameter	E^{2}C-stage1	E^{2}C-stage2	GRPO
Batch Size	256	256	128
Overlong Buffer Length	4096	4096	4096
Maximum Response Length	8192	8192	8192
Learning Rate	1.0×10^{-6}	1.0×10^{-6}	1.0×10^{-6}
Mini-batch Size for GRPO Updates	32	32	32
KL Loss Coefficient β	0.001	0	0
Rollout Number k	32	8	8
Temperature	1.3	1.0	1.0
Training Epochs	1	1	5
Clip ratio (ε)	0.2	0.2	0.2

Table 6: Hyperparameters for E^{2}C-RL and GRPO Training

ensuring that the output adheres to the expected length constraints.

A.3 Details of the Algorithm

Algorithm of E^{2}C-SFT Data Generation Algorithm 1 formalizes the causal data pipeline introduced in Section 3.2.1.

Algorithm 1 E^{2}C-SFT Data Generation

$\mathcal{D}_{\text{synth}} \leftarrow \emptyset$
for each question q **do**
  solution $\leftarrow \text{Model}_{\text{base}}(q)$
  exploration $\leftarrow \text{Summarize}(\text{solution})$
  prompt $\leftarrow$ "Given question: q. Follow exploration: exploration. Execute step-by-step:"
  execution $\leftarrow \text{Model}_{\text{base}}(\text{prompt})$
  $\mathcal{D}_{\text{synth}} \leftarrow \mathcal{D}_{\text{synth}} \cup \{(q, (\text{exploration,execution}))\}$
end for
return $\mathcal{D}_{\text{synth}}$

Algorithm of Exploration-Focused SFT (EF-SFT) Data Generation Algorithm 2 is a formal and detailed description for EF-SFT Data Generation.

Algorithm 2 EF-SFT Data Generation

Require: Base E^{2}C dataset $\mathcal{D}_{\text{base}}$
Require: Domain-specific dataset $\mathcal{D}_{\text{domain}}$
Require: Mixing ratio $\alpha \in [0, 1]$,Target Dataset size n_{target}
Ensure: EF-SFT training dataset $\mathcal{D}_{\text{EF-SFT}}$

$\mathcal{D}_{\text{explore}} \leftarrow \emptyset$
for each example $(q, a) \in \mathcal{D}_{\text{domain}}$ **do**
  Extract exploration segment: $e \leftarrow \text{ExtractExploration}(a)$
  $\mathcal{D}_{\text{explore}} \leftarrow \mathcal{D}_{\text{explore}} \cup \{(q, e)\}$
end for
$n_{\text{base}} \leftarrow \alpha \times n_{\text{target}}$ ▷ $\alpha\%$ from base dataset
$n_{\text{explore}} \leftarrow (1 - \alpha) \times n_{\text{target}}$ ▷ $(1 - \alpha)\%$ from exploration data
$\mathcal{D}^{\text{sub}}_{\text{base}} \leftarrow \text{Subsample}(\mathcal{D}_{\text{base}}, n_{\text{base}})$
$\mathcal{D}^{\text{sub}}_{\text{explore}} \leftarrow \text{Subsample}(\mathcal{D}_{\text{explore}}, n_{\text{explore}})$
$\mathcal{D}_{\text{EF-SFT}} \leftarrow \mathcal{D}^{\text{sub}}_{\text{base}} \cup \mathcal{D}^{\text{sub}}_{\text{explore}}$
return $\mathcal{D}_{\text{EF-SFT}}$

Algorithm of E^{2}C Test-Time Scaling Algorithm 3 formalizes the shared explore-select-

execute-refine template introduced in Section 3.3.

Algorithm 3 E^{2}C Test-Time Scaling

Generate or expand K exploration segments: $\{\pi_1, \pi_2, \ldots, \pi_K\}$
Select a compact candidate set Π^* via judge, clustering, tree pruning, or fallback plans
for each selected exploration $\pi_i^* \in \Pi^*$ **do**
    Generate deterministic execution: $a_i \leftarrow \text{Execute}(\pi_i^*)$
    if execution stalls and the budget remains **then**
        Revise or resample the exploration: $\pi_i^* \leftarrow \text{Refine}(\pi_i^*)$
        Retry execution with the revised exploration
    end if
end for
Let a_{final} be the selected answer or the weighted-vote winner
return a_{final}

A.4 Test-Time Scaling Experimental Details

This section provides a detailed description of the experimental setup for the test-time scaling comparison presented in Table 3, following the template in Algorithm 3.

Objective and General Setup The primary goal was to test whether E^{2}C methods that scale the Exploration stage outperform full-chain baselines that scale complete reasoning chains. All methods used the same checkpoint (**Qwen3-8B+E^{2}C-(SFT+RL)**) to ensure a fair comparison of inference strategies. Diverse sampling steps used temperature 0.9. Performance is reported as Pass@1 accuracy, and cost is measured by the average total generated tokens per question.

Full-Chain Baselines

- **Greedy CoT** (N=1): A single reasoning chain generated with greedy decoding. Serves as the zero-budget reference.
- **Self-Consistency (SC)**: For each budget level $N \in \{4, 8, 16, 32\}$, N full independent CoT chains were sampled and majority-voted.
- **Tree-of-Thoughts (ToT)**: ToT tree expansion was applied to full reasoning chains. Branch factor 2, depth $\lceil \log_2 N \rceil$; leaf nodes were completed to full solutions and majority-voted.
- **ReAct** (N=1): A single-pass ReAct-style attempt that interleaves reasoning with self-correction; serves as a single-loop iterative reference.

E^{2}C Methods All E^{2}C variants use the model's Exploration/Execution format; $K \in \{4, 8, 16, 32\}$ controls the exploration budget.

- **E^{2}C-Select (Self LM-Judge)**: K exploration plans were sampled; the model then acted as a judge to select the single most promising plan; one deterministic execution was generated from that plan.
- **E^{2}C-ToT**: ToT-style branching was applied solely over the Exploration stage (branch factor 2, depth $\lceil \log_2 K \rceil$). The resulting plan leaves were clustered with K-Means (M=3 clusters; `all-mpnet-base-v2` embeddings); the centroid plan of each cluster was executed and answers were aggregated by cluster-size-weighted majority vote.
- **E^{2}C-ReAct Loop**: K explorations were sampled; the self LM-judge selected an initial plan; it was executed deterministically. If no valid boxed answer was found or the execution showed looping/hallucination patterns, the model revised the plan using the failed execution as feedback (or switched to an unused candidate plan). The loop was capped at 3 refinement rounds; each refinement call used at most 1024 tokens.

A.5 Entropy Visualization of Different RL Settings and Analysis

In this part, we visualize the entropy dynamics and the accuracy on the AIME'24 benchmark during RL training. The results demonstrate that applying our token-weighting coefficient $\lambda_{i,t}$ to exploration tokens facilitates a rapid drop in entropy and a better performance improvement, as shown in **Fig. 3**. This is achieved by effectively amplifying high-quality plans while suppressing poor ones.

A.6 Prompt Details

E^{2}C-SFT Dataset Construction prompt

Exploration Phase Prompt The following prompt is used to extract the high-level exploration plan from the reasoning process:

Role: You are an expert problem-solver.
Task: Distill a complex reasoning

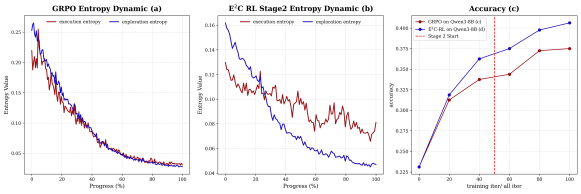

Figure 3: A comparison of training dynamics on the AIME'24 benchmark. The application of our token-weighting coefficient $\lambda_{i,t}$ (b) facilitates faster entropy reduction and superior performance improvement compared to the baseline without it (a).

process into a clear, actionable plan.

Input:

- **Problem:** `<question>`
- **Reasoning Process:** `<content>`

Output Requirements:

1. **Format:** Present the summary as a numbered list (e.g., 1., 2., 3.).
2. **Content:** For each step, describe only the essential action to be taken (e.g., "Calculate X," "Verify Y"). Be concise and prescriptive.
3. **Focus:** Omit explanations, justifications, or intermediate conclusions.

Goal: Create a high-level plan that is easy to follow and execute.

Execution Phase Prompt The following prompt is used to generate the detailed execution steps based on the exploration plan:

Role: You are a meticulous problem solver.
Task: Solve the given question by strictly following the provided guideline, showing all detailed reasoning.

Input:

- **Question:** `<question>`
- **Guideline:** `<content>`

Output Requirements:

1. Follow the guideline exactly, numbering each step accordingly (e.g., 1., 2., ...).
2. Do not include any content outside the solution steps.
3. Begin from Step 1, expanding each step with necessary calculations and logical reasoning.
4. Conclude by placing the final answer within a '`\boxed{}`' environment.

Important: Ensure every mathematical or logical operation is explicitly shown.

EF-SFT dataset Construction prompt The following prompt is used to extract the exploration part for EF-SFT dataset in medical domain.

Role: You are a professional doctor.
Task: Summarize the diagnostic reasoning process into a concise, actionable guideline.

Input:

- **Question:** `<question>`
- **Reasoning Process:** `<content>`

Output Requirements:

1. **Structure:** Present the summary as a numbered list (1., 2., ...), starting directly with the first step.
2. **Conciseness:** Use no more than 5 steps. Each step must be under 15 words and state only the critical objective (e.g., "Assess cardiac function").
3. **Focus:** Highlight the most critical diagnostic step. Omit all explanations, justifications, or unrelated content.

Goal: Create a concise and accurate diagnostic plan focused on key actions.

LLM-Combination Prompt To enable the model to select the most promising exploration plan, we use the following prompt. The model is instructed to act as an impartial judge, evaluating the provided plans based on their clarity, correctness, and likelihood of leading to a successful solution.

Role: You are an expert mathematical reasoner and an impartial judge. Your task is to evaluate several proposed plans for solving a given math problem and identify the single best one.

Input:

- **Problem:** <problem>
- **Candidate Plans:** A numbered list of K exploration plans. Plan 1: $< exploration_1 >$ Plan 2: $< exploration_2 >$ ... Plan K: $< exploration_K >$

Instructions:

1. Carefully analyze the problem and each of the K candidate plans.
2. Assess the plans based on their logical soundness, potential for success, and efficiency.
3. Select the single best plan that is most likely to lead to a correct and complete solution.

Output Format: Output only the full text of the single best plan you have selected. Do not add any extra commentary, explanation, or formatting.

Adherence Judge Prompt The following prompt is used to evaluate whether an execution strictly adheres to the provided exploration plan.

Role: You are a rigorous evaluator. Your task is to judge if the execution strictly follows the exploration plan.

Input:

- **Question:** $< question >$
- **Exploration (PLAN):** $< exploration >$
- **Execution:** $< execution >$

Evaluation Criteria: Provide a final score of 0, 0.5, or 1.0 based on the following:

- **1.0 (Strictly Adheres):** The execution follows the exploration's logic and key steps exactly. All critical reasoning or calculations in the exploration are present and correctly implemented in the execution.
- **0.5 (Partially Adheres):** The execution addresses the main goal but deviates in specifics: it may skip non-critical steps, change the order of minor steps, or contain minor logical gaps while reaching a correct conclusion.
- **0.0 (Does Not Adhere):** The execution ignores major exploration steps, follows a different approach, contains significant logical errors, or fails to implement the exploration's core intent.

Output Format: First, provide a concise rationale comparing the exploration and execution. Then, output the score strictly as:
[RATIONALE]: <your analysis>
[SCORE]: <0.0, 0.5, or 1.0>

A.7 E^{2}C Reasoning Examples

Example 1(Math, from AMC23): Meeting Problem

Input Question Cities A and B are 45 miles apart. Alicia lives in A and Beth lives in B. Alicia bikes towards B at 18 miles per hour. Leaving at the same time, Beth bikes toward A at 12 miles per hour. How many miles from City A will they be when they meet? Provide the final answer in the boxed.

Exploration Phase 1. Identify the speeds and initial distance between the two cities. 2. Determine the combined speed at which the distance between Alicia and Beth is closing. 3. Calculate the time it takes for them to meet by dividing the initial distance by their combined speed. 4. Use the time to calculate the distance Alicia travels from City A.

Execution Phase

1. **Identify the speeds and initial distance between the two cities.** Distance between City A and City B: 45 miles. Alicia's speed: 18 miles per hour (towards B). Beth's speed: 12 miles per hour (towards A)

2. **Determine the combined speed at which the distance between Alicia and Beth is closing.** Since they are moving towards each other, their speeds add up:

$$\text{Combined speed} = 18 + 12 = 30 \text{ miles per hour}$$

3. **Calculate the time it takes for them to meet by dividing the initial distance by their combined speed.** The time until they meet is:

$$\text{Time} = \frac{\text{Distance}}{\text{Combined speed}} = \frac{45}{30} = 1.5 \text{ hours}$$

4. **Use the time to calculate the distance Alicia travels from City A.** Alicia's distance from City A when they meet is:

$$\text{Distance} = \text{Speed} \times \text{Time} = 18 \times 1.5 = 27 \text{ miles}$$

Final Answer

$$\boxed{27}$$

Example 2 (Math, from AIME'24): Table Tennis Tournament Problem

Input Question In a table tennis tournament every participant played every other participant exactly once. Although there were twice as many right-handed players as left-handed players, the number of games won by left-handed players was 40% more than the number of games won by right-handed players. (There were no ties and no ambidextrous players.) What is the total number of games played? Provide the final answer in the boxed.

Exploration Phase 1. Define variables for the number of left-handed and right-handed players. 2. Express the total number of games played in terms of the total number of players. 3. Determine the total number of games won by left-handed and right-handed players. 4. Set up an equation based on the given relationship between wins. 5. Solve the equation to find the total number of players. 6. Calculate the total number of games played.

Execution Phase

1. **Define variables for the number of left-handed and right-handed players.** Let L be the number of left-handed players, and R be the number of right-handed players. Given: $R = 2L$

2. **Express the total number of games played in terms of the total number of players.** Total players: $L + R = 3L$ Total games: $\binom{3L}{2} = \frac{3L(3L-1)}{2}$

3. **Determine the total number of games won by left-handed and right-handed players.** Let W_L be wins by left-handed players, W_R be wins by right-handed players. Total wins: $W_L + W_R = \frac{3L(3L-1)}{2}$ Given: $W_L = 1.4W_R$

4. **Set up an equation based on the given relationship between wins.** Substitute $W_L = 1.4W_R$ into total wins: $1.4W_R + W_R = \frac{3L(3L-1)}{2}$ $2.4W_R = \frac{3L(3L-1)}{2}$

5. **Solve the equation to find the total number of players.** Try $L = 3$: $W_R = \frac{3\times3\times(9-1)}{4.8} = \frac{72}{4.8} = 15$ $W_L = 1.4 \times 15 = 21$ Total wins: $15 + 21 = 36$

6. **Calculate the total number of games played using the total number of players.** Total games: $\frac{3\times3\times(9-1)}{2} = \frac{72}{2} = 36$

Final Answer

$$\boxed{36}$$

Example 3 (Medical,From MMLU-Anatomy): Urinary System Question

Input Question Which of the following best describes the structure that collects urine in the body? A. Bladder B. Kidney C. Ureter D. Urethra Please reasoning step-by-step. Provide the final answer in the boxed.

Exploration Phase 1. Identify the function of each organ listed in the options. 2. Determine which organ is responsible for storing urine. 3. Eliminate options that do not fit the function of collecting urine. 4. Select the organ that best matches the description of collecting urine.

Execution Phase

1. **Identify the function of each organ listed in the options:**
   - **A. Bladder:** Hollow, muscular organ that stores urine until excretion
   - **B. Kidney:** Filters blood and produces urine, but does not store urine
   - **C. Ureter:** Transports urine from kidneys to bladder, does not store urine
   - **D. Urethra:** Carries urine from bladder to outside body, does not store urine

2. **Determine which organ is responsible for storing urine:** The bladder is the primary organ responsible for urine storage. It expands as it fills with urine and contracts during urination.

3. **Eliminate options that do not fit the function of collecting urine:**
   - Kidney (B): Produces urine but does not store it
   - Ureter (C): Transports urine but does not store it

- Urethra (D): Excretes urine but does not store it
- Bladder (A): Only organ designed for urine collection and storage

4. **Select the organ that best matches the description of collecting urine:** The bladder is the structure that collects and stores urine in the body.

Final Answer

$\boxed{A}$

Example 4 (Test-Time Scaling, from AIME'24): Table Tennis Tournament Problem This example demonstrates how E^{2}C test-time scaling operates over compact exploration plans rather than full reasoning chains, using the same input question from Example 2.

Input Question In a table tennis tournament every participant played every other participant exactly once. Although there were twice as many right-handed players as left-handed players, the number of games won by left-handed players was 40% more than the number of games won by right-handed players. (There were no ties and no ambidextrous players.) What is the total number of games played? Provide the final answer in the boxed.

Step 1: Sample K Exploration Plans (K=4) First, the E^{2}C model is prompted to generate K=4 diverse, high-level exploration plans for the problem.

- **Plan 1** 1. Let L be the number of left-handed players, the total games are C(3L, 2). 2. Let W_L be wins by lefties and W_R be wins by righties. 3. Solve this system for L, then find the total games.
- **Plan 2** 1. The total number of players must be a multiple of 3, let's call it 3L. The total games played is C(3L, 2). 2. Let's test small integer values for L (L=1, 2, 3...) and check if the resulting total games can be split into wins for left- and right-handed players satisfying the 40% more condition.
- **Plan 3** 1. Assume the number of wins is proportional to the number of players. Let right-handed players have W_R wins. 2. Left-handed players have half the number of players, so they should have W_L wins. 3. Set up W_L = 1.4 * W_R and solve based on the total number of games.
- **Plan 4** 1. Let the number of wins by right-handed players be W_R. Then the wins by left-handed players is 1.4 * W_R. 2. The total number of games is 2.4 * W_R. The total number of games is also given by C(3L, 2). 3. Set C(3L, 2) = 2.4 * W_R and find an integer solution for L.

Method A: E^{2}C-Select (Self LM-Judge) The four plans above, along with the original question, are fed into the model with the Self LM-Judge prompt. The model evaluates the plans and selects the most robust and direct strategy.

1. **Selection:** The Self LM-Judge identifies Plan 1 as the most comprehensive and logically sound approach, as it correctly sets up the system of equations from first principles.
2. **Execution:** A single execution is performed, conditioned only on Plan 1. This execution proceeds exactly as detailed in Example 2, arriving at the correct answer.

Final Answer (Self LM-Judge): $\boxed{36}$

Method B: E^{2}C-ReAct Loop This feedback-driven method executes the selected plan, and only returns to the exploration stage if execution fails to produce a valid final answer.

1. **Initial Selection:** The Self LM-Judge again selects Plan 1 as the initial exploration because it directly sets up the algebraic relationship between left- and right-handed wins.
2. **Execution Attempt:** The model executes the selected plan deterministically. If the execution reaches a valid boxed answer, the loop stops immediately.
3. **Exploration Refinement on Failure:** If execution stalls, loops, or fails to produce a boxed answer, the recent execution trace is summarized as feedback. The model then revises the exploration plan rather than generating a new full solution. For example, it may add the missing constraint that the total number of games equals both $\binom{3L}{2}$ and $W_L + W_R$.
4. **Fallback to Unused Plans:** If the revised plan is too similar to the failed one, the method switches to an unused candidate plan that avoids the stuck point.

5. **Retry Execution:** The revised or replacement exploration is executed again. In this example, the corrected plan yields the same final result:
   - Let $W_L = 1.4W_R$ and $W_L + W_R = \binom{3L}{2}$.
   - Enforce integrality of the number of players and wins.
   - Solve the resulting constraints to obtain the total number of games.

Final Answer (E^{2}C-ReAct Loop): $\boxed{36}$

A.8 Pure Prompt-based E^{2}C

We product an experiment with pure prompt-based E^{2}C on Qwen3-8B. For each problem we first sample K independent *exploration* traces by prompting the model K times with a short exploration prompt; each exploration is a concise (2–4 short sentence) reasoning sketch that does not contain the final answer. We then combine the K explorations into a single execution prompt (providing the problem and the numbered explorations) and ask the model to produce one final *Execution:* section that computes the final answer. Performance is reported as pass@5 for different values of K.The results are much worse than the E^{2}C model with E^{2}C-(SFT+RL), which demonstates that a prompt engeneering is not enough.

Exploration prompt The following prompt was used to generate each individual exploration (one exploration per model call).

> **Role:** You are a careful math problem solver.
>
> **Input:**
>
> - **Problem:** `<problem>`
>
> **Instructions:**
>
> - Produce exactly one short reasoning sketch (an *exploration*) that helps approach the problem.
> - The exploration must be concise (about 2–4 short sentences).
> - Do **not** produce the final answer in this call.
> - Stop immediately after the single exploration text and do not append any extra commentary, labels, or formatting.
>
> **Output format:** A single short exploration paragraph (2–4 short sentences) and nothing else.

Table 7: Pass@5 accuracy (%) for different numbers of sampled explorations K.

Dataset	$K = 2$	$K = 3$	$K = 4$	$K = 5$
MATH500	84.4	83.2	84.0	84.0
AIME24	26.7	33.0	36.7	26.7
AIME25	23.3	30.0	30.0	26.7

Execution prompt The following prompt was used to synthesize the K independently sampled explorations into a final execution.

> **Role:** You are a careful math problem solver.
>
> **Input:**
>
> - **Problem:** `<problem>`
> - **Explorations:**
> Exploration 1: `<exploration 1>`
> Exploration 2: `<exploration 2>`
> ⋮
> Exploration $\{K\}$: `<exploration K>`
>
> **Instructions:**
>
> - Learn from the provided $\{K\}$ numbered explorations and combine their useful reasoning to compute the final answer.
> - Produce a single **Execution:** section that carries out the computation and presents the final answer.
> - Stop immediately after the final answer. Do not append extra commentary, explanations, or any additional text beyond the required Execution section and the answer.