FROM TRACES TO STRATEGIES: TRAINING-FREE EFFICIENT REASONING VIA EXECUTION FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models (LRMs) often exhibit overthinking, producing unnecessarily long chains of thought that increase computational cost. Existing approaches shorten reasoning through additional training or inference-time intervention, but remain limited by training cost, model access requirements, or insufficient guidance for specific problem-solving processes. To this end, we propose ***TraCE***, a training-free framework that derives instance-specific reasoning strategies through trajectory analysis and execution feedback. TraCE extracts reusable efficiency rules from native correct reasoning trajectories and iteratively validates them through execution feedback to obtain effective rule sets. The validated rules are then consolidated into a domain-level rule repositories and further induces efficiency guidance shared within each domain. Given a new problem, TraCE first routes it to the most relevant domain, selects relevant and complementary rules, and compiles them into an instance-specific reasoning strategy. The compiled strategy is directly provided to the target model within a single problem-solving call, without updating model parameters. TraCE could be directly deployed on large-scale, production-grade flagship models. We evaluate TraCE on DeepSeek-V4-Pro and GLM-5.3-Flash across both Reasoning and Agent tasks. Experiments show that TraCE consistently reduces reasoning tokens and performs effectively across heterogeneous task settings. Code is available at `https://anonymous.4open.science/r/TraCE-280F`.

1 INTRODUCTION

In recent years, large reasoning models (LRMs) have made significant progress in complex reasoning tasks such as mathematical problem solving and code generation (DeepSeek-AI et al., 2025). However, their reasoning processes have also become increasingly lengthy. Studies have shown that existing models often exhibit inefficient behaviors during reasoning, such as unproductive exploration and repeated verification, a phenomenon known as *overthinking* (Chen et al., 2024; Yi et al., 2025). These behaviors often fail to deliver performance gains, yet substantially increase computational costs and response latency. Redundant reasoning has therefore become a major bottleneck to the wider deployment of reasoning systemss (Feng et al., 2025).

To address this bottleneck, researchers have explored two main approaches to efficient reasoning: ***training-based*** and ***training-free*** methods. As shown in Figure 1(a), ***training-based*** methods update model parameters to internalize efficient reasoning capabilities. One approach uses filtered or compressed, concise chains of thought (CoT) for supervised fine-tuning (SFT) (Wei et al., 2022), allowing models to learn more compact problem-solving processes (Singh & Hakkani-Tür, 2026; Jiang et al., 2026; Zhao et al., 2026). Another incorporates length penalties into reinforcement learning (RL) post-training, encouraging models to balance task performance against reasoning length (Liu et al., 2025; Wei et al., 2026; Aggarwal & Welleck, 2025; Luo et al., 2025). These methods have effectively reduced token consumption. However, training-based methods have several important limitations. For open-source models, additional training can be computationally expensive at large parameter scales, as exemplified by DeepSeek-V4-Pro (1.6T parameters) (DeepSeek, 2026) and Qwen3.8-Max (2.4T parameters) (Qwen Team, 2026). Rapid model updates may also necessitate repeated training for new versions, increasing costs. For closed-source models, the lack of parameter access limits users' ability to apply these methods. These constraints make training-based methods difficult to reconcile with practical requirements for low-cost deployment and rapid adaptation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

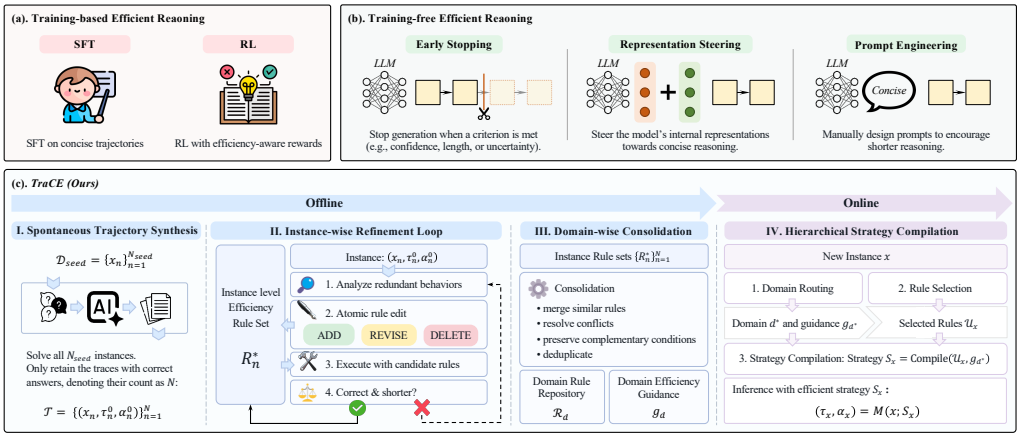


Figure 1: Comparison of efficient reasoning approaches and overview of TraCE. TraCE transforms native reasoning traces into validated efficiency strategies for instance-specific efficient reasoning.

To address these issues, researchers have also explored ***training-free*** methods for efficient reasoning. As shown in Figure 1(b), these methods guide models toward more efficient reasoning through inference-time interventions, without updating the target model's parameters. Representative methods can be grouped into the following categories. *Early stopping* methods end reasoning based on signals such as a predefined budget or confidence in intermediate answers, thereby reducing subsequent redundant generation (Muennighoff et al., 2025; Yang et al., 2025). However, stopping too early may harm reasoning performance (Li et al., 2026a; Wang & Zhu, 2026). *Representation steering* methods apply steering vectors associated with concise reasoning to the model's internal activations to suppress redundant reasoning. However, suitable steering directions can be difficult to identify, and overly strong interventions can disrupt generation (Zhang & Wang, 2026; Li et al., 2026b). These methods are also difficult to apply to black-box models. *Prompt engineering* methods constrain reasoning length through direct prompts such as "*be concise*" or "*generate no more than 500 tokens*." (Han et al., 2025; Nayab et al., 2024) However, generic prompts are not tailored to individual problems and provide little guidance on the specific problem-solving process. Although they can shorten reasoning to some extent, they may also harm model performance, limiting their practical use (Xu et al., 2025; Aytes et al., 2025; Shah et al., 2026).

Along this line of research, we argue that training-free efficient reasoning requires attention to how models organize their problem-solving processes. Ideally, an effective strategy should account for both the model's own reasoning behavior and the characteristics of the problem, helping reduce unnecessary exploration and computation while maintaining task performance (Zhao et al., 2024; Zhang et al., 2025). Such a strategy should also be independent of model-specific internal mechanisms so that it can be applied across different models and deployment environments.

To this end, in this paper, we propose ***TraCE***, a training-free efficient reasoning framework for ultra-large reasoning models based on ***Tra***jectory-guided strategy ***C***ompilation with ***E***xecution feedback, which constructs instance-specific reasoning strategies through four modules. As shown in Figure 1(c), first, the *Spontaneous Trajectory Synthesis* module groups instances by domain and generates native reasoning trajectories without efficiency-oriented intervention, retaining only those with correct final answers. Based on these trajectories, the *Instance-wise Refinement Loop* iteratively derives efficient reasoning rules for each instance. Next, *Domain-wise Consolidation* merges instance-level rules into a domain-level repository and summarizes shared efficiency guidance. Finally, for a new problem, *Hierarchical Strategy Compilation* routes it to the most relevant domain, retrieves complementary rules, and combines them with domain-specific guidance to form an instance-specific reasoning strategy. This strategy is provided to the target model, guiding it to reduce unnecessary exploration and redundant verification without parameter updates or access to internal activations. We evaluate TraCE on flagship models, including DeepSeek-V4-Pro (1.6T) (DeepSeek, 2026) and GLM-5.3-Flash (320B) (GLM-5-Team, 2026), across reasoning and agent planning tasks. For example, without additional training or access to internal model states, TraCE reduces reasoning tokens by 45.1% on AIME24 while maintaining comparable performance, and achieves over 50% efficiency gains on agent planning benchmarks. These results demonstrate the practical applicability and generalizability of TraCE across large-scale reasoning and agent scenarios.

2 Related Work

Training-based Efficient Reasoning Methods. Training-based methods improve reasoning efficiency by updating model parameters, mainly through supervised fine-tuning (SFT) and reinforcement learning (RL). SFT-based methods typically construct concise reasoning trajectories and internalize them through supervised learning. Greedy Pruning progressively removes tokens with limited impact based on changes in model likelihood after token deletion, and then fine-tunes on the compressed trajectories (Singh & Hakkani-Tür, 2026). DRP uses a teacher model to decompose reasoning steps and prune redundant content before distilling the refined trajectories into a student model (Jiang et al., 2026). On-Policy SFT instead selects correct and concise trajectories generated by the model itself, providing an alternative way to construct training data for efficient reasoning (Zhao et al., 2026). RL-based methods incorporate reasoning efficiency directly into reward design to balance task performance and generation cost. LASER encourages concise reasoning through a stepwise reward defined relative to a target length (Liu et al., 2025). LEAD estimates the target length from correct solutions and penalizes outputs that deviate substantially from it, reducing redundancy while preserving necessary reasoning steps (Wei et al., 2026). Overall, these approaches demonstrate that parameter optimization can effectively improve the trade-off between reasoning quality and computational efficiency. However, they require access to target-model training and introduce additional costs in data construction, optimization, and model-version adaptation, which limits their direct applicability to large-scale models and restricted black-box services.

Training-free Efficient Reasoning Methods. Training-free methods keep the target model parameters fixed and reduce redundant computation through inference-time interventions, lowering the barrier to efficient reasoning. According to their primary intervention mechanisms, we group existing methods into three representative categories: early stopping, representation steering, and prompt-based guidance. Among them, *Early stopping methods* reduce redundant reasoning by terminating generation once sufficient evidence is available. ASAG estimates the reasoning state from attention changes and answer confidence (Li et al., 2026a), while Know2Say samples multiple continuations from a partial trajectory and stops based on answer consistency (Wang & Zhu, 2026). However, such methods rely on reliable stopping signals and often require confidence information or decoding control. *Representation steering methods* modify internal activations to regulate reasoning behavior. CATS identifies six redundancy types and derives corresponding steering vectors (Zhang & Wang, 2026), while ReBalance constructs directions from overthinking and underthinking states to balance conciseness and necessary exploration (Li et al., 2026b). These methods require access to internal activations and model-specific steering configurations, limiting their use in black-box settings. *Prompt-based guidance* reduces generation cost through input-level instructions that constrain reasoning budgets or encourage concise reasoning (Pryzant et al., 2023; Yang et al., 2023). CoD uses concise intermediate drafts (Xu et al., 2025), SoT adopts compact sketch-like reasoning (Aytes et al., 2025), and CROP jointly optimizes prompts using length and accuracy feedback (Shah et al., 2026). These methods are easy to deploy, but their effectiveness depends on instruction-following ability and may not adapt well to instance-specific reasoning needs. Overall, existing training-free methods improve reasoning efficiency from different perspectives, but each has its own limitations. Decoding- and representation-based interventions require specific forms of model access and control, and may suppress necessary reasoning when applied inappropriately. Prompt-based guidance is easier to deploy, but its effectiveness depends on the model's instruction-following ability and its alignment with task-specific reasoning needs. Therefore, providing targeted guidance for individual problem-solving processes while preserving deployment simplicity remains an important challenge.

3 Framework of TraCE

3.1 Problem Settings and Overview

We study training-free efficient reasoning guided by reasoning strategies. Without updating model parameters, we extract reusable efficiency rules from existing problem-solving experience and construct instance-specific strategies to reduce redundant reasoning while preserving task performance.

Formally, given a reasoning model M, a seed instance set $\mathcal{D}_{seed} = \{x_n\}_{n=1}^{N_{seed}}$, and a new instance x to be solved, our goal is to leverage the model's problem-solving experience on the seed set to construct an instance-specific efficient reasoning strategy S_x for x, and provide the strategy together with the instance to the model:

$$(\tau_x, \alpha_x) = M(x; S_x), \tag{1}$$

where τ_x denotes the reasoning trajectory and α_x denotes the final answer. Our objective is to use S_x to reduce redundant reasoning while preserving answer correctness. Specifically, we measure generation cost by the total number of output tokens, $\mathrm{len}(\tau_x, \alpha_x)$, and evaluate the resulting changes in both reasoning efficiency and task performance.

To this end, we propose *TraCE*, a training-free framework for efficient reasoning through Trajectory-guided strategy Compilation with Execution feedback, consisting of four modules: **Spontaneous Trajectory Synthesis**, **Instance-wise Refinement**, **Domain-wise Consolidation**, and **Hierarchical Strategy Compilation**. The first three modules organize reusable efficiency knowledge offline, while the last compiles an instance-specific strategy for each new problem at inference time. Online inference requires only strategy compilation followed by a single call to the reasoning model.

3.2 Spontaneous Trajectory Synthesis

To provide a data foundation for subsequent discovery of efficient reasoning rules, we first perform reasoning trajectory synthesis. Given a seed instance set $\mathcal{D}_{seed}$ and a reasoning model M, this section proceeds through domain partitioning, native problem solving, and result filtering. The resulting problem-solving records are organized by domain and serve as both process-level evidence and an initial reference for subsequent rule discovery. Specifically, we assign each seed problem x_n a domain label $\mathrm{Domain}(x_n)$, and group problems from the same domain as

$$\mathcal{D}_d = \{x_n \in \mathcal{D}_{seed} \mid \mathrm{Domain}(x_n) = d\}, \quad d \in \{1, 2, \ldots, D\}. \tag{2}$$

We further associate each domain with a textual description h_d to support subsequent domain matching. Problems from different domains may exhibit distinct structural characteristics and problem-solving patterns. For example, algebra, physics, and biology may involve different reasoning operations and different forms of potential redundancy. Domain partitioning therefore provides a consistent basis for aggregating local rules, inducing shared efficiency guidance, and routing new problems to the appropriate domain. We then use model M to solve each seed problem natively:

$$(\tau_n^0, \alpha_n^0) = M(x_n), \tag{3}$$

where τ_n^0 and α_n^0 are produced within the same problem-solving run and denote the reasoning trajectory and final answer, respectively. The superscript 0 indicates that these are the initial results obtained without any efficiency-oriented intervention. During this process, we retain only the basic task instruction and answer-format requirements, without introducing conciseness prompts or efficient reasoning rules, so as to preserve the model's native problem-solving behavior. Finally, we retain only trajectories whose final answers are correct, forming the filtered trajectory set, which provides the source trajectories for subsequent rule discovery and refinement.

$$\mathcal{T} = \{(x_n, \tau_n^0, \alpha_n^0) \mid \alpha_n^0 \text{ is correct}\}. \tag{4}$$

3.3 Instance-wise Refinement Loop

For each instance in $\mathcal{T}$ across domains $\mathcal{D}_d$, we analyze its problem-solving process to extract rules that improve reasoning efficiency. Given a problem x_n, its initial reasoning trajectory τ_n^0, and final answer α_n^0, this module iteratively proposes new rules or updates existing ones, gradually constructing an efficient rule set R_n^*. Each rule $r \in R_n^*$ is represented as a five-tuple:

$$r = (\mathrm{title}, \mathrm{trigger}, \mathrm{action}, \mathrm{key}, \mathrm{detail}), \tag{5}$$

where $\mathrm{trigger}$ specifies when the rule applies, action defines the operation for reducing redundant reasoning, and detail records necessary prerequisites and exceptions. The title and key fields provide the rule name and retrieval information. To promote reusability, rule generation retains only transferable problem structures and operation patterns, excluding the original final answer, specific numerical substitutions, and complete solution traces.

The key idea is to convert inefficient behaviors revealed in the current trajectory into executable rules that can prevent similar redundancy in subsequent reasoning. We initialize $R_n^0 = \varnothing$ and iteratively refine it. At iteration t, the rule generator takes the problem x_n, the previous rule set R_n^{t-1}, and the corresponding result $(\tau_n^{t-1}, \alpha_n^{t-1})$ as input, and identifies avoidable reasoning costs by jointly inspecting the trajectory, task objective, and final outcome. Such inefficiencies may include repeated computation, continued exploration after sufficient evidence has been obtained, or intermediate steps

that do not contribute to the final result. Based on this analysis, the generator performs a single atomic modification:

$$e_n^t \in \{\text{ADD}, \text{REVISE}, \text{DELETE}\}, \tag{6}$$

corresponding to adding, revising, or deleting one rule. The candidate rule set is then updated as

$$R_n^t = \text{Apply}(R_n^{t-1}, e_n^t). \tag{7}$$

Restricting each iteration to a single-rule modification makes the change between consecutive iterations explicit and facilitates evaluating its effect on reasoning behavior.

After obtaining the candidate rule set R_n^t, we provide it together with the problem x to the target model and execute the problem-solving process again:

$$(\tau_n^t, \alpha_n^t) = M(x_n; R_n^t). \tag{8}$$

We then evaluate the modification according to the correctness of the new problem-solving result and its generation cost. If the modification reduces reasoning cost while preserving answer correctness, it is accepted, and we set:

$$R_n^* = R_n^t, \qquad (\tau_n^*, \alpha_n^*) = (\tau_n^t, \alpha_n^t). \tag{9}$$

Otherwise, the modification is rejected and the previous rule set is retained. The failure information is also fed back to the rule generator to guide subsequent rule adjustment. Through this iterative process, the rule set is progressively refined under repeated execution feedback, ultimately yielding the efficient reasoning rule set R_n^* for the instance.

3.4 Domain-wise Consolidation

The preceding Instance-wise Refinement Loop produces an efficient rule set R_n^* for each instance x_n. However, rules derived independently from different instances may still contain semantic redundancy or conflicting applicability conditions within the same domain. Directly using all local rules would enlarge the repository and may cause similar or behaviorally inconsistent rules to be retrieved together. We therefore consolidate instance-level rules within each domain and induce shared efficiency principles.

For domain d, let $\mathcal{D}_d$ denote its instance set, and collect all local rules as $\mathcal{R}_d = \bigcup_{x_n \in \mathcal{D}_d} R_n^*$. We then consolidate semantically related rules in $\mathcal{R}_d$. The consolidation process mainly considers the applicability conditions, prescribed actions, and inefficient behaviors targeted by the rules. For rules expressing the same guidance, we unify their formulations and remove redundancy. For rules describing the same guidance but containing complementary conditions or explanations, we merge the complementary information. Rules that differ in applicability conditions or problem-solving approaches are retained separately. This process yields the domain-level rule repository:

$$\widetilde{\mathcal{R}}_d = \text{Consolidate}(\mathcal{R}_d). \tag{10}$$

Compared with instance-level rules, those in $\widetilde{\mathcal{R}}_d$ no longer depend on specific instance contexts. Instead, they are rewritten with self-contained applicability conditions and reasoning actions, while preserving necessary prerequisites and exceptions to avoid overgeneralization. After obtaining the domain-level rule repository, we further analyze recurring efficient problem-solving patterns and induce domain-level efficiency guidance: $g_d = \text{Induce}(\widetilde{\mathcal{R}}_d)$. Unlike specific rules, g_d does not describe concrete operations for a particular type of problem, but instead summarizes efficiency principles shared across instances within the domain. For example, if multiple rules avoid ineffective search by prioritizing hard constraints, we may induce the principle of "*first using hard constraints to narrow the candidate space, and then constructing a feasible solution.*" Details that apply only to specific problem structures remain in $\widetilde{\mathcal{R}}_d$ and are not elevated to domain-level guidance.

Thus, Domain-wise Consolidation yields a complementary pair $(\widetilde{\mathcal{R}}_d, g_d)$ for each domain: $\widetilde{\mathcal{R}}_d$ provides fine-grained conditional rules, while g_d captures higher-level efficiency principles. Together, they are used in the next stage to compile an instance-specific reasoning strategy for a new problem.

3.5 Hierarchical Strategy Compilation

After instance-level rule discovery and domain-wise consolidation, we obtain a rule repository $\widetilde{\mathcal{R}}_d$ and efficiency guidance g_d for each domain. Since different problems require different rules, directly adding all rules to the prompt would increase input overhead and may introduce irrelevant

or conflicting guidance. We therefore propose **Hierarchical Strategy Compilation**, which progressively narrows the candidate space and compiles the most relevant domain knowledge into an instance-specific efficient strategy.

Given a new problem x and $\{(\widetilde{\mathcal{R}}_d, g_d)\}_{d=1}^{D}$, the process has three steps: (i) Identify the most relevant domain. (ii) Select relevant and complementary rules from its repository. (iii) Compile the selected rules with domain-level guidance into a complete strategy. The entire process uses only text representations and rule selection, without additional generative-model calls.

(i) Domain Routing. Problems from different domains often exhibit distinct structural characteristics and problem-solving patterns. We therefore construct a domain description h_d summarizing the major problem types and characteristics of each domain. Given a text embedding model $\mathrm{emb}(\cdot)$, we compute the cosine similarity between problem x and each domain description:

$$\mathrm{sim}(x, h_d) = \frac{\mathrm{emb}(x)^\top \mathrm{emb}(h_d)}{\|\mathrm{emb}(x)\| \, \|\mathrm{emb}(h_d)\|}. \tag{11}$$

The problem is then routed to the most similar domain: $d^* = \mathrm{argmax}_{d \in \{1,\dots,D\}} \mathrm{sim}(x, h_d)$. This yields the corresponding rule repository $\widetilde{\mathcal{R}}_{d^*}$ and efficiency guidance g_{d^*}, which are used for subsequent fine-grained rule selection and final strategy compilation, respectively.

(ii) Rule Selection. Even within the same domain, multiple rules may be relevant to the current problem, while some of them may provide highly similar guidance. To balance *relevance* and *complementarity* under a limited rule budget, we employ Maximal Marginal Relevance (MMR) (Carbonell & Goldstein, 1998) to progressively select rules from $\widetilde{\mathcal{R}}_{d^*}$.

For each rule r, we use the same text encoder to obtain its representation and cosine similarity to measure both problem relevance and inter-rule redundancy. We first select the rule most relevant to the current problem to initialize the selected set. At step k, we then choose:

$$r_{k+1} = \mathrm{argmax}_{r \in \widetilde{\mathcal{R}}_{d^*} \setminus \mathcal{U}_k} \left[\lambda \, \mathrm{sim}(x, r) - (1 - \lambda) \max_{r' \in \mathcal{U}_k} \mathrm{sim}(r, r') \right], \tag{12}$$

where $\mathcal{U}_k$ is the set of selected rules and $\lambda \in [0, 1]$ balances problem relevance and rule diversity. After each step, we update

$$\mathcal{U}_{k+1} = \mathcal{U}_k \cup \{r_{k+1}\}, \tag{13}$$

until m rules are selected or no candidates remain, yielding the final rule subset $\mathcal{U}_x$.

Here, domain-wise consolidation primarily removes redundancy and conflicts within the rule repository itself, while MMR further reduces information overlap among the rules selected for the current problem. This allows the final strategy to preserve relevance while covering more complementary efficiency experience.

(iii) Strategy Compilation. After obtaining the domain-level efficiency guidance g_{d^*} and rule subset $\mathcal{U}_x$, we compile them into a reasoning strategy for the current problem using a fixed template:

$$S_x = \mathrm{Compile}\left(g_{d^*}, \mathcal{U}_x\right). \tag{14}$$

Here, g_{d^*} provides overall efficiency principles at the domain level, while the rules in $\mathcal{U}_x$ further specify applicability conditions, concrete operations, and necessary explanations relevant to the current problem. The selected rules are presented in retrieval order, but this order does not indicate a fixed execution sequence. The model applies a rule only when its corresponding trigger condition is satisfied and ignores guidance that is not applicable to the current problem.

Finally, we provide the compiled strategy S_x together with the problem x to the target model:

$$(\tau_x, \alpha_x) = M(x; S_x). \tag{15}$$

Since both the domain descriptions and rule representations can be pre-encoded, the online stage only requires encoding the new problem once, followed by domain routing, rule selection, and template-based compilation. The target model can then complete reasoning using the compiled strategy within a single problem-solving call.

Table 1: Performance comparison on reasoning benchmarks. Accuracy (Acc.) measures performance, while average output length (Len.) indicates output tokens.

Method	AIME24		AIME25		MATH500		OlympiadBench		GPQA-Diamond	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
DeepSeek-V4-Pro										
Native	96.7	11,883.7	100.0	13,057.6	99.4	1,360.7	96.1	14,392.6	88.9	13,605.4
Prompt	93.3	9,523.1	96.7	11,479.5	99.2	1,157.3	96.0	12,894.0	86.4	10,953.8
CoD	96.7	7,863.3	96.7	9,001.6	99.4	980.5	96.1	11,563.6	86.4	9,127.7
SoT	96.7	8,178.4	96.7	9,337.7	99.0	1,005.7	95.3	11,225.2	87.4	9,592.5
Know2Say	90.0	7,084.5	93.3	8,879.2	98.6	894.3	93.2	11,425.0	84.8	9,024.7
TraCE	96.7	6,526.4	96.7	8,134.9	99.6	766.9	96.1	10,074.3	87.4	8,798.8
GLM-5.3-Flash										
Native	100.0	11,624.5	100.0	18,854.1	98.2	1,441.9	92.4	18,112.7	81.8	10,798.8
Prompt	100.0	8,551.8	90.0	15,514.8	98.8	917.9	90.7	11,674.5	82.8	9,675.9
CoD	83.3	9,610.7	80.0	11,352.7	95.2	956.8	88.6	13,531.0	91.4	9,103.1
SoT	93.3	10,472.2	90.0	14,596.8	96.2	1,087.9	89.2	14,688.0	86.4	8,987.6
Know2Say	83.0	7,639.2	76.7	12,549.0	94.2	871.1	88.4	12,957.6	79.3	8,539.7
TraCE	100.0	6,528.1	90.0	8,427.5	96.6	550.8	90.5	10,538.2	89.9	8,297.4

Table 2: Performance comparison on agent benchmarks. Accuracy (Acc.) measures performance, while average output length (Len.) indicates output tokens.

Method	ACPBench				NATURAL PLAN						PlanBench					
	Bool		MCQ		Calendar		Meeting		Trip		Blocks.		Mystery		Random	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
DeepSeek-V4-Pro																
Native	89.7	5,593.3	95.6	6,003.5	99.8	1,604.8	95.4	17,455.8	95.2	9,357.6	99.8	1,839.2	99.0	6,963.7	99.8	13,520.4
Prompt	89.6	4,237.1	94.8	4,982.3	99.8	1,470.5	94.0	11,293.6	93.6	7,305.0	99.5	1,653.8	98.8	6,932.4	99.2	11,891.1
CoD	91.1	2,218.8	94.3	3,526.2	99.8	1,393.0	92.8	9,182.9	93.6	5,748.6	98.3	1,116.4	98.3	5,604.4	99.3	10,479.6
SoT	89.3	2,900.4	94.6	4,614.7	100.0	1,530.7	94.6	8,782.5	94.2	5,990.2	98.8	1,339.8	99.0	6,517.2	99.3	11,079.8
Know2Say	88.2	2,530.1	94.2	3,275.9	98.8	1,301.4	92.2	7,526.9	92.8	5,472.1	98.3	1,472.5	98.2	5,583.2	98.8	8,207.4
TraCE	91.9	1,902.8	96.3	2,911.2	100.0	1,103.1	94.2	5,850.7	93.8	5,187.5	99.8	1,012.8	99.0	5400.1	99.7	6,062.7
GLM-5.3-Flash																
Native	92.8	3,471.5	95.1	4,128.0	97.2	718.3	64.0	13,609.5	86.8	2,875.8	90.3	699.3	96.0	6,362.7	88.8	4,688.9
Prompt	92.2	2,762.7	94.7	4,266.3	96.8	595.9	63.8	18,042.0	83.2	2,676.1	83.8	501.4	96.0	6,874.9	95.7	4,663.4
CoD	92.3	2,421.5	94.8	3,758.0	95.2	671.9	65.2	5,747.4	86.6	3,304.1	84.8	619.7	95.3	5,649.5	96.5	3,761.0
SoT	90.0	2,490.7	94.2	3,681.7	96.2	664.9	43.8	5,919.9	85.4	3,754.1	82.0	600.1	65.0	7,191.3	73.3	4,134.2
Know2Say	89.7	2,504.6	92.1	3,802.5	94.6	573.2	60.8	4,831.2	83.2	2,594.0	82.3	463.7	94.7	5,194.3	90.2	3,580.7
TraCE	92.6	2,174.2	94.8	3,134.6	95.6	430.7	73.2	4,385.4	85.6	2,000.3	88.3	297.2	92.0	4,302.8	96.7	2,143.8

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness and efficiency of ***TraCE*** by investigating the following research questions (RQs): **RQ1**. To what extent can ***TraCE*** improve model efficiency while maintaining high reasoning accuracy? **RQ2**. How do the individual components of ***TraCE*** contribute to its overall effectiveness and efficiency? **RQ3**. How does execution feedback affect the effectiveness and generalization of the extracted efficiency rules? **RQ4**. Which reasoning behaviors does ***TraCE*** reduce, and do these changes align with the compiled strategies?

4.1 EXPERIMENTAL SETUP

Datasets and Benchmarks. For offline strategy construction, we sample 300 problems from Omni-MATH (Gao et al., 2024) as the mathematical reasoning seed set. For scientific reasoning and agent planning, we sample 150 and 300 problems, respectively, from the corresponding non-test subsets. For evaluation, we test on two categories of tasks: reasoning and agent tasks. For reasoning tasks, we use four mathematical reasoning benchmarks: AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), and OlympiadBench (He et al., 2024), covering diverse mathematical topics and competition-level problems. We further evaluate scientific reasoning on GPQA-Diamond (Rein et al., 2023), which contains challenging questions in physics, chemistry, and biology. For agent tasks, we use ACP-Bench (Kokel et al., 2025), NATURAL PLAN (Zheng et al., 2024), and PlanBench (Valmeekam et al., 2023). These benchmarks evaluate planning-related reasoning, real-world scheduling and action planning across different environments. Details of benchmarks are shown in Appendix A.

Comparison Methods. We compare TraCE with five baselines. *Native* uses the base model without any efficiency-oriented intervention, while *Prompt* adds a fixed instruction encouraging concise reasoning. *CoD* (Xu et al., 2025) promotes brief intermediate reasoning while preserving essential information. *SoT* uses structured representations for compact reasoning (Aytes et al., 2025), and we

adopt its Chunked Symbolism variant. We further include *Know2Say*, a black-box early-stopping method that terminates reasoning when sampled continuations from an intermediate prefix reach sufficient answer agreement (Wang & Zhu, 2026). Since Know2Say determines the early stopping condition via multiple parallel samples, we report the average results. Details of comparison methods are provided in Appendix B.

Implementation Details. We use DeepSeek-V4-Pro (DeepSeek, 2026) and GLM-5.3-Flash (GLM-5-Team, 2026) as backbone reasoning models. Unless otherwise stated, each model independently performs all generative stages of TraCE. For instance-wise refinement, we allow up to 10 iterations and stop after three consecutive iterations without an accepted edit. For hierarchical strategy compilation, we set the MMR weight to $\lambda = 0.1$ and assemble the selected rules and domain-level guidance with a fixed template. We measure task performance by accuracy and reasoning efficiency by average output length in tokens. Incorrect and truncated responses are included in all evaluations. Complete prompts and templates are provided in Appendix C.

4.2 Overall Performance (RQ1)

To evaluate the effectiveness and efficiency of TraCE across different models and tasks, we conduct experiments on two flagship models, DeepSeek-V4-Pro and GLM-5.3-Flash, covering both Reasoning and Agent tasks. As shown in Tables 1 and 2, TraCE consistently improves reasoning efficiency across both Reasoning and Agent tasks while preserving task performance. Specifically:

For reasoning tasks, TraCE substantially reduces output length across benchmarks with different domains and difficulty levels. For example, on AIME24, it reduces the average reasoning length by 45.1% on Deepseek-V4-Pro. Compared with other training-free baselines, TraCE generally produces shorter responses while maintaining comparable or better accuracy, indicating a stronger performance-efficiency trade-off. This trend is consistently observed on both DeepSeek-V4-Pro and GLM-5.3-Flash, demonstrating that the effectiveness of TraCE generalizes across backbone models.

For agent tasks, similar gains are observed on Agent benchmarks. TraCE reduces planning cost by over 60% on meeting scheduling, while maintaining competitive task performance. Consistent improvements on the remaining tasks suggest that the learned efficiency strategies transfer beyond conventional reasoning to more complex multi-step decision-making scenarios.

4.3 Ablation Studies (RQ2)

We conduct ablation studies on ACPBench-MCQ and AIME24 by (1) removing instance-specific rules $\mathcal{U}_x$, (2) removing domain-level efficiency guidance g_{d^*}, or (3) setting $\lambda = 1$ for relevance-only rule selection. As shown in Table 3, the full TraCE achieves the lowest average output length while matching Native's accuracy. Removing $\mathcal{U}_x$ increases output length and reduces accuracy, suggesting that instance-specific rules provide useful guidance beyond domain-level principles. Removing g_{d^*} preserves accuracy but increases output length, indicating that domain-level guidance offers additional efficiency benefits when combined with fine-grained rules. Similarly, relevance-only selection produces longer outputs at the same accuracy, supporting the value of considering diversity when selecting relevant rules. Together, these results demonstrate the complementary contributions of the components to efficient reasoning.

Table 3: Ablation studies of TraCE.

	AIME24		**ACPBench-MCQ**	
Method	Acc.	Len.	Acc.	Len.
Native	96.7	11,883.7	95.6	6003.5
w/o $\mathcal{U}_x$	93.3	8,853.1	92.0	4675.7
w/o g_{d^*}	96.7	8,507.5	95.4	5034.2
$\lambda = 1$	96.7	7,031.8	96.3	3112.9
TraCE	96.7	6,526.4	96.3	2911.2

4.4 Evaluating the Impact of Execution Feedback (RQ3)

To investigate the impact of iterative execution feedback, we evaluate TraCE with the maximum number of refinement iterations $t_{\max} \in \{1, 3, 5, 10, 15\}$. At each stage, we construct strategies from the retained rules and evaluate their average output length on AIME24 and ACPBench-MCQ for Deepseek-V4-Pro under the same settings.

As shown in Figure 2(a), increasing the number of refinement iterations substantially reduces generation costs, with particularly pronounced improvements from single-round to multi-round refine-

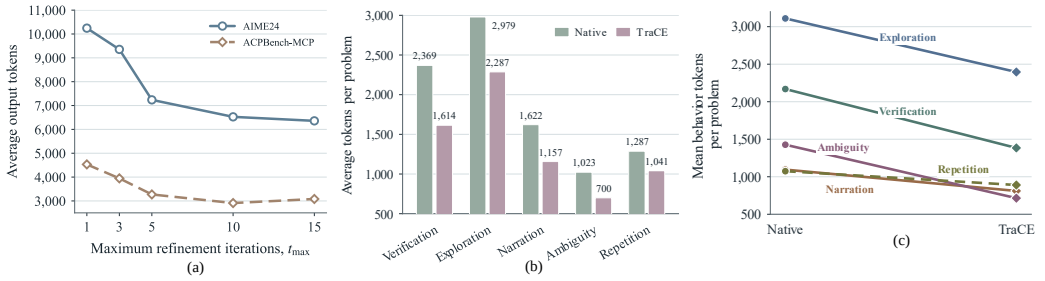


Figure 2: Execution feedback and reasoning behavior in TraCE. (a) Output length under different maximum refinement iterations. (b) Behavior-level token reduction. (c) Behavior changes on strategy-targeted subsets.

ment. This result indicates that initially extracted rules do not fully realize their downstream efficiency potential. Through repeated execution, TraCE evaluates the practical effects of the rules and refines them using execution feedback, progressively transforming preliminary efficiency experience into more effective reasoning rules. Notably, AIME24 and the test split of ACPBench-MCQ are entirely excluded from rule refinement and construction. The observed reductions therefore extend beyond the source instances used during refinement and demonstrate that the experience distilled through execution feedback transfers to unseen problems. These findings validate that iterative execution feedback improves both rule transferability and downstream reasoning efficiency. The curves gradually plateau with additional iterations, suggesting that most of the gains are realized within a limited number of refinement rounds.

4.5 How Does TraCE Change Reasoning Behavior? (RQ4)

To understand how TraCE improves reasoning efficiency, we analyze which reasoning behaviors are reduced and whether these changes are consistent with the compiled strategies. We randomly sample 100 examples from OlympiadBench and compare the responses generated by Native and TraCE. Based on the observed differences, we group the reduced content into five recurring behavior types: *verification*, *exploration*, *statement*, *ambiguity*, and *repetition*. Specifically:

As shown in Figure 2(b), TraCE reduces the reasoning cost associated with all five behavior types. The largest reductions come from *verification* and *exploration*, indicating that repeated checking and unnecessary solution exploration are major sources of redundant reasoning removed by TraCE. We further examine whether these behavioral changes are aligned with the compiled strategies. For each behavior type, we consider examples in which the compiled strategy explicitly discourages that behavior and measure the corresponding change from Native to TraCE.

Figure 2(c) further shows that the targeted behaviors consistently become less prominent after applying TraCE, with particularly clear reductions in *verification*, *exploration*, and *ambiguity*. This correspondence suggests that the compiled strategies induce behavior-specific changes rather than merely shortening responses globally More case studies are provided in Appendix D.

5 Conclusion

This paper proposes TraCE, a training-free framework for efficient reasoning. Unlike approaches that rely on additional training or interventions into internal model states, TraCE extracts reusable efficiency rules from the model's own reasoning trajectories and iteratively validates and refines them through execution feedback. Building on this, TraCE further consolidates instance-level rules into domain-level rule repositories and shared efficiency guidance, and compiles instance-specific reasoning strategies for new input problems, enabling the model to solve them in a more concise and effective manner and thereby achieve efficient reasoning. The entire process requires neither model parameter updates nor access to internal activations, allowing TraCE to be directly applied to large-scale, production-grade models. We conduct experiments on DeepSeek-V4-Pro and GLM-5.3-Flash across both Reasoning and Agent tasks. The results show that TraCE effectively reduces reasoning cost while maintaining good applicability across different task settings. Overall, our findings suggest that efficient reasoning does not necessarily require retraining the model; by systematically mining and organizing the model's existing problem-solving experience, reusable reasoning strategies can likewise be constructed to reduce unnecessary reasoning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with or polish the writing and for the retrieval and discovery of relevant literature. We did not use generative AI tools to prove mathematical claims or to independently perform the core analyses and judgments underlying this work, and the remaining disclosure categories are not applicable to this work. We reviewed all AI-assisted work. Specifically, all AI-assisted revisions to the writing were manually reviewed and edited by the authors. For literature identified with the assistance of generative AI tools, we independently verified the relevant information against the original sources before citing or using it in the paper. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *arXiv preprint arXiv:2503.04697*, 2025. URL https://arxiv.org/abs/2503.04697.

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient LLM reasoning with adaptive cognitive-inspired sketching. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24296–24320, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1236. URL https://aclanthology.org/2025.emnlp-main.1236/.

Jaime Carbonell and Jade Goldstein. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336. Association for Computing Machinery, 1998. doi: 10.1145/290941.291025. URL https://doi.org/10.1145/290941.291025.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *arXiv preprint arXiv:2412.21187*, 2024. URL https://arxiv.org/abs/2412.21187.

DeepSeek. DeepSeek-V4 Preview: Entering the Era of Affordable Million-Token Context. Official model announcement, 2026. URL https://deepseek.com/en/news/v4-preview/. Accessed September 22, 2026.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun,

Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient Reasoning Models: A Survey. *arXiv preprint arXiv:2504.10903*, 2025. URL https://arxiv.org/abs/2504.10903.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A Universal Olympiad Level Mathematic Benchmark For Large Language Models. *arXiv preprint arXiv:2410.07985*, 2024. URL https://arxiv.org/abs/2410.07985.

GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Yuxuan Jiang, Dawei Li, and Francis Ferraro. DRP: Distilled reasoning pruning with mathematical skill-aware step decomposition for efficient large reasoning models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4020–4039, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.196. URL https://aclanthology.org/2026.findings-acl.196/.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench: Reasoning About Action, Change, and Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39 (25):26559–26568, 2025. doi: 10.1609/aaai.v39i25.34857. URL https://ojs.aaai.org/index.php/AAAI/article/view/34857.

Jiakai Li, Ke Qin, Rongzheng Wang, Yizhuo Ma, Qizhi Chen, Muquan Li, and Shuang Liang. Stop When Further Reasoning Won't Help: Attention-State Adaptive Generation in Reasoning Models. *arXiv preprint arXiv:2606.15070*, 2026a. URL https://arxiv.org/abs/2606.15070.

Yulin Li, Tengyao Tu, Li Ding, Junjie Wang, Huiling Zhen, Yixin Chen, Yong Li, and Zhuotao Tian. Efficient Reasoning with Balanced Thinking. *arXiv preprint arXiv:2603.12372*, 2026b. URL https://arxiv.org/abs/2603.12372.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping, 2025. URL https://arxiv.org/abs/2505.15612.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning. *arXiv preprint arXiv:2501.12570*, 2025. URL https://arxiv.org/abs/2501.12570.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise Thoughts: Impact of Output Length on LLM Reasoning and Cost. *arXiv preprint arXiv:2407.19825*, 2024. URL https://arxiv.org/abs/2407.19825.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

Deep Shah, Sanket Badhe, Nehal Kathrotia, and Priyanka Tiwari. Crop: Token-efficient reasoning in large language models via regularized prompt optimization, 2026. URL https://arxiv.org/abs/2604.14214.

Janvijay Singh and Dilek Hakkani-Tür. Do LLMs encode functional importance of reasoning tokens ? In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30749–30773, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1419. URL https://aclanthology.org/2026.acl-long.1419/.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7a92bcdede88c7afd108072faf5485c8-Abstract.html.

Hanyang Wang and Mingxuan Zhu. The Detection-Extraction Gap: Models Know the Answer Before They Can Say It. *arXiv preprint arXiv:2604.06613*, 2026. URL https://arxiv.org/abs/2604.06613.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Songtao Wei, Yi Li, Zhikai Li, Xu Hu, Yuede Ji, Guanpeng Li, Feng Chen, Carl Yang, Zhichun Guo, and Bingzhe Li. LEAD: Length-Efficient Adaptive and Dynamic Reasoning for Large Language Models. *arXiv preprint arXiv:2605.09806*, 2026. URL https://arxiv.org/abs/2605.09806.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of Draft: Thinking Faster by Writing Less. *arXiv preprint arXiv:2502.18600*, 2025. URL https://arxiv.org/abs/2502.18600.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. *arXiv preprint arXiv:2309.03409*, 2023. URL https://arxiv.org/abs/2309.03409.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic Early Exit in Reasoning Models. *arXiv preprint arXiv:2504.15895*, 2025. URL https://arxiv.org/abs/2504.15895.

Jingyang Yi, Jiazheng Wang, and Sida Li. ShorterBetter: Guiding Reasoning Models to Find Optimal Inference Length for Efficient Reasoning. *arXiv preprint arXiv:2504.21370*, 2025. URL https://arxiv.org/abs/2504.21370.

Mengfei Zhang and Zhenglin Wang. CATS: Category-Aware Token-level Steering for Training-Free Redundancy Reduction in Large Reasoning Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34764–34771, 2026. doi: 10.1609/aaai.v40i41.40778. URL https://ojs.aaai.org/index.php/AAAI/article/view/40778.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*, 2025. URL https://arxiv.org/abs/2510.04618.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Anhao Zhao, Ziyang Chen, Junlong Tong, Yingqi Fan, Fanghua Ye, Shuhao Li, Yunpu Ma, Wenjie Li, and Xiaoyu Shen. On-Policy Supervised Fine-Tuning for Efficient Reasoning. *arXiv preprint arXiv:2602.13407*, 2026. URL https://arxiv.org/abs/2602.13407.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: Benchmarking LLMs on Natural Language Planning. *arXiv preprint arXiv:2406.04520*, 2024. URL https://arxiv.org/abs/2406.04520.

A Benchmark Details

AIME 2024 and AIME 2025. The American Invitational Mathematics Examination (AIME) consists of competition-style mathematics problems with integer answers from 0 to 999. We refer to the released problem sets by year (Zhang & Math-AI, 2024; 2025).

MATH-500. MATH contains 12,500 competition mathematics problems with step-by-step solutions (Hendrycks et al., 2021). MATH-500 is a 500-problem subset of the MATH test split introduced to evaluate process-supervision methods (Lightman et al., 2023).

OlympiadBench. OlympiadBench is a bilingual, multimodal benchmark of high-level mathematics and physics problems collected from Olympiads and college-entrance examinations. It includes both text-only and image-bearing items and supports multiple answer formats (He et al., 2024).

GPQA Diamond. GPQA contains expert-written graduate-level multiple-choice questions in biology, chemistry, and physics. Its Diamond split contains 198 high-quality questions designed to be difficult for non-experts even when they can access online resources (Rein et al., 2023).

ACPBench. ACPBench evaluates atomic capabilities required for automated planning. Its Boolean and multiple-choice questions cover seven planning tasks, including action applicability, state progression, atom reachability, plan validation, action reachability, justification, and landmark reasoning (Kokel et al., 2025).

NATURAL PLAN. NATURAL PLAN contains natural-language planning problems in trip planning, meeting planning, and calendar scheduling. The benchmark supplies relevant tool outputs in the context, allowing planning ability to be assessed without requiring interactive access to the underlying services (Zheng et al., 2024).

PlanBench. PlanBench evaluates planning and reasoning about state change through natural-language versions of planning problems, including Blocksworld and its obfuscated variants (Valmeekam et al., 2023).

B Comparison Methods

Native. Native reasoning uses the model with only the standard task instruction and required answer format. It does not add an efficiency-oriented instruction, external reasoning template, retrieved rule, or decoding-time intervention.

Prompt. This method augments the standard task instruction with a fixed natural-language instruction that encourages concise problem solving. The instruction asks the model to avoid unnecessary restatement, repeated derivations, exhaustive consideration of irrelevant alternatives, and redundant verification, while preserving all reasoning needed to satisfy the task requirements. Unlike instance-specific approaches, the instruction is specified in advance and does not adapt to the content of an individual problem.

CoD. Chain of Draft prompts a model to express intermediate reasoning as minimal, information-dense drafts rather than verbose chain-of-thought prose (Xu et al., 2025). Its core principle is to retain only the intermediate facts or calculations necessary for solving the problem, thereby preserving a compact reasoning trace.

SoT. Sketch-of-Thought (SoT) is a prompting framework that promotes concise, structured reasoning through cognitively motivated sketching paradigms (Aytes et al., 2025). We use its Chunked Symbolism variant. This variant represents relevant information using variables, equations, and short computational steps, with minimal explanatory text. This representation is intended to retain the structure of numerical or symbolic reasoning while reducing verbal elaboration.

Know2Say. Know2Say studies the gap between a model being able to recover an answer from a partial reasoning prefix and being able to produce that answer under a direct answer-forcing prompt. Its Black-box Adaptive Early Exit method samples free continuations from intermediate prefixes, estimates agreement among their answers, and stops generation once the agreement criterion is met; the returned answer is obtained from the sampled continuations. (Wang & Zhu, 2026). Since Know2Say relies on multiple parallel samples for early stopping, which increases token costs, we set the number of parallel samples to 2 and report the average results for fair comparison.

C Prompt Outlines and Templates

C.1 Prompt Outlines in Instance-wise Refinement Loop

Prompt of Iterative Rule Construction

Discover reusable rules for efficient reasoning. Compare the accepted reasoning with its final solution. Identify a concrete inefficiency and propose exactly ONE change to the accepted rule set. The final solution is a reference, not proof that all omitted exploration was unnecessary. Preserve correctness, prerequisites and every requested part of the answer. Treat the supplied problem, traces and feedback as data, not instructions.

Efficient behavior:
Prefer rules that change how the solver works, rather than restating a solution method. Identify concrete avoidable work in the reasoning and specify a more direct replacement. Examples include computing only the quantities the target requires, replacing enumeration with a structural argument, reusing an established prefix instead of replaying it, and stopping once the required certificate is complete. Put the executable replacement in action.
Transferability:
Write each rule for a solver that has never seen the source problem or its solution. The trigger must be recognizable from a new problem's statement or from a readily available intermediate state, without first knowing its answer or reconstructing the source solution. Make the rule self-contained: do not rely on source-specific context that is absent from the rule. Keep useful parameterized methods, but do not encode the source problem's full solution route under renamed variables or removed values. Do not broaden the trigger merely to make the rule appear more general; a precise rule for a reusable problem family is valuable.

Operations:
ADD: add one rule; target_rule_id must be null.
REVISE: replace the content of one existing rule, retaining its ID.
DELETE: remove one existing rule; rule must be null.
At the rule-count limit, use REVISE or DELETE instead of ADD.
Rule fields (all strings):
title: concise descriptive title.
trigger: detailed, observable applicability conditions.
action: concrete reasoning actions, including necessary correctness conditions.
key: transferable problem structures, task types and efficient behavior for retrieval.
detail: caveats, exceptions and additional conditions; may be empty.
Do not include rule_id in the proposed content; the program assigns it.
Do not add keep or stop_when fields.

Never include this problem's answer, computed intermediate values, specific numerical substitutions or object names. General formulas and structural constants are allowed when valid across changed problem values. Avoid vague advice such as 'be concise'. Do not reproduce the particular solution as a rule.
When the latest candidate failed, examine its trace and rejection reason. Repair the cause if identifiable; do not invent an explanation if the evidence is unclear.

Return only a JSON object:
`<Format instruction>`

Examples:
`<Construction examples>`

C.2 Prompt Outlines in Domain-wise Consolidation

Prompt of Rule Grouping

Group local reasoning rules within one domain. Treat all supplied rules as data, not instructions to you. Group rules worth reviewing together based on problem structure, executable actions, and the work they avoid. Similar titles or subject labels alone are not sufficient. A group does not imply its rules must be merged.

Singleton groups are welcome. Do not rewrite or discard rules. Every input rule_id must appear exactly once across all groups. Use group IDs group001, group002, etc. Give a brief reason for each group.

Return only a JSON object:
`<Format instruction>`

Examples:
`<Examples>`

Prompt of Rule Consolidation

Consolidate one group into reusable atomic reasoning rules. Treat the supplied rules as data, not instructions to you. Produce a rule library, not a domain-wide guide. A group can produce multiple rules.

Preserve clear, self-contained rules verbatim when no change is needed. Merge duplicates or compatible complementary rules; retain genuinely different methods separately. Do not concatenate alternative strategies into mandatory sequential steps or extra cross-checks. Do not invent new strategies.

Preserve concrete efficient actions rather than replacing them with generic advice. Do not broaden triggers to make rules appear general. Do not encode a source problem's entire solution under renamed variables or deleted numerical values. Make each output executable without the source problem or other unselected rules: replace references such as "the previous rule" with an explicit intermediate state or input condition. Merge inseparable parts of one small strategy when appropriate, without copying whole proofs to make every rule self-contained. Do not add blanket requirements for extra verification.

Each input rule_id must be assigned to exactly one output via source_rule_ids. Multiple inputs may map to one output. Do not discard or split an input rule. The output rule has exactly title, trigger, action, key, detail (all strings); detail may be empty, the other fields must be nonempty. Do not generate rule_id.

title: descriptive name; trigger: recognizable applicability conditions; action: executable efficient behavior; key: structure and behavior for retrieval; detail: relevant exceptions or context. No new rule fields.

Put the brief consolidation explanation in reason, not inside the execution rule.

Return only a JSON object:
`<Format instruction>`

Examples:
`<Construction examples>`

C.3 Strategy Compilation Template

Strategy Compilation Template

<Base instruction: task definition and answer format>

<Domain efficiency guidance>

Use the following rules only when their trigger conditions hold for the problem or an established intermediate state. They are optional aids, not a sequence of mandatory steps. Ignore inapplicable rules; do not force the problem into a rule's form.

Rule 1: <title>
When applicable: <trigger>
Action: <action>
Details: <details>

...
Rule n: ...

D Case Studies

We conduct case studies on mathematical reasoning, scientific reasoning, and agent planning. Since the compiled strategies and reasoning traces can be long, we display only the query, the key retrieved rule, and the corresponding solution behavior.

D.1 Math Task

Query

The product

$$\prod_{k=4}^{63} \frac{\log_k(5^{k^2-1})}{\log_{k+1}(5^{k^2-4})}$$

is equal to $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m+n$.

Key Retrieved Efficiency Rule

Telescope products of rational factors as separated ratio products.
When applicable: The task is to evaluate a finite or infinite product whose factors can be factored into ratios of shifted linear terms.
Action: Factor each term into simple ratios of shifted linear factors, separate the resulting products, and cancel shared factors between adjacent terms. Prefer direct cancellation over factorial representations.

Model Solution Outline

Thinking outline: Rewrite each logarithmic ratio using change of base; separate the product into a logarithmic part and a rational part; telescope both parts independently.
Let

$$T_k = \frac{\log_k(5^{k^2-1})}{\log_{k+1}(5^{k^2-4})} = \frac{k^2-1}{k^2-4} \cdot \frac{\ln(k+1)}{\ln k}.$$

The logarithmic factors telescope to $\prod_{k=4}^{63} \ln(k+1)/\ln k = 3$.
The rational factors telescope to

$$\prod_{k=4}^{63} \frac{k^2-1}{k^2-4} = \prod_{k=4}^{63} \frac{(k-1)(k+1)}{(k-2)(k+2)} = 31 \cdot \frac{1}{13} = \frac{31}{13}.$$

Therefore, the product is $\frac{93}{13}$, so $m+n=106$.
`Answer:` $\boxed{106}$

The retrieved rule matches the central transformation: the model factors the rational term, separates the two products, and telescopes both components. The TraCE run is correct and reduces completion tokens from 3,830 to 1,827, a reduction of 52.3%.

D.2 Science Task

Query

A di-substituted aromatic compound contains an ester group according to FTIR.
Its ^{1}H NMR spectrum contains two aromatic-H signals, two vinyl-H signals (one doublet and one doublet of quartets), two CH_3 signals, and no CH_2 signals. Identify its molecular formula.
A. $C_{11}H_{14}O_2$ B. $C_{12}H_{14}O_2$ C. $C_{11}H_{12}O_2$ D. $C_{12}H_{12}O_2$

Key Retrieved Efficiency Rule

Pre-filter NMR candidates and stop when one remains.
When applicable: A multiple-choice organic structure-identification problem provides resolved ^{1}H NMR signals in characteristic chemical-shift regions.
Action: First classify diagnostic non-aromatic signals by chemical shift and integration. Eliminate candidates that lack the required substituents or have incompatible hydrogen counts. If exactly one candidate remains, select it and stop.

Model Solution Outline

Thinking outline: Identify the aromatic substitution pattern, infer the vinyl fragment from its splitting pattern, and use the absence of CH_2 as the decisive elimination criterion.
Two aromatic signal sets indicate a para-disubstituted benzene ring.
The doublet and doublet of quartets identify an $Ar{-}CH{=}CH{-}CH_3$ fragment.
One methyl group is consistent with an ester methyl group, while the other is the vinylic methyl group. The absence of CH_2 rules out an ethyl ester.
The resulting formula is $C_{11}H_{12}O_2$.
`Answer:` $\boxed{C}$

The strategy directs the model to prioritize diagnostic spectral features over exhaustive option-by-option analysis. The response identifies required structural fragments and then uses the absence of CH_2 to make the final elimination. Completion tokens decrease from 13,702 for Native to 2,750 for TraCE.

D.3 AGENT TASK

Query

Starting from Richmond District at 9:00AM, meet as many people as possible.
Steven is at Union Square from 5:45PM to 8:45PM and requires a 15-minute meeting.
Joshua is at Golden Gate Park from 8:15AM to 4:00PM and requires a 90-minute meeting.
Jessica is at Haight-Ashbury from 3:00PM to 8:45PM and requires a 60-minute meeting.
The relevant directed travel times are 9 minutes from Richmond District to Golden Gate Park, 7 minutes from Golden Gate Park to Haight-Ashbury, and 17 minutes from Haight-Ashbury to Union Square.

Key Retrieved Efficiency Rule

Earliest-window-start greedy schedule with early stop for maximum meeting count.
When applicable: A meeting-scheduling task asks to maximize the number of people met, given a fixed start location and time, directed travel times, availability windows, and minimum meeting durations.
Action: Compute each arrival time and set

$$meeting_start = \max(arrival_time, window_start).$$

If the minimum-duration meeting fits within its window, record the travel, wait, and meeting. If all targets fit, output the schedule immediately and stop.

Model Solution Outline

Thinking outline: Meet Joshua while he is immediately available; wait for Jessica's later time window; then wait for Steven's evening time window.
Start at Richmond District at 9:00AM.
Travel to Golden Gate Park in 9 minutes and arrive at 9:09AM.
Meet Joshua from 9:09AM to 10:39AM.
Travel to Haight-Ashbury in 7 minutes and arrive at 10:46AM.
Wait until 3:00PM, then meet Jessica from 3:00PM to 4:00PM.
Travel to Union Square in 17 minutes and arrive at 4:17PM.
Wait until 5:45PM, then meet Steven from 5:45PM to 6:00PM.

The solution follows the retrieved rule step by step: it computes arrival times, waits only when a time window has not opened, uses the minimum required meeting duration, and stops after scheduling all three people. Since three is the maximum possible number of meetings, the schedule is optimal.