BEHAVIOR CUE REASONING: MONITORABLE REASONING IMPROVES SAFETY AND EFFICIENCY THROUGH OVERSIGHT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning in Large Language Models (LLMs) poses a challenge for oversight as many misaligned behaviors can surface only in downstream actions once reasoning has concluded. To address this, we introduce Behavior Cue Reasoning for making LLM reasoning more controllable and monitorable. Behavior Cues are special token sequences that a model is trained to emit immediately before specific implicit and explicit behaviors, acting as dual purpose signal and control levers. Our experiments reveal that a Behavior Cue Reasoning model has equal to greater performance as the base model, allows for steerable reasoning through external enforcement of Behavior Cues, and improves the monitorability of reasoning for external oversight monitors. When leveraged by an oracle monitor in an environment where excessive constraint violations results in failure, Behavior Cues allows for the recovery of safe actions from up to 83% of reasoning traces that would otherwise end with the proposal of an unsafe action. When applying early-stopping efficiency methods, Behavior Cues see a similar token-saving, accuracy trade off without the need for otherwise costly answer probes. Through evaluation across two model families and three domains, we show that Behavior Cue Reasoning improves reasoning monitorability and controllability with no cost to performance. More broadly, our work progresses scalable oversight by demonstrating how the monitored model itself can be trained to reason in a more tractable to oversight.

1 INTRODUCTION

For difficult tasks, LLMs are trained to search within their token space by generating reasoning traces to refine their response before answering. These traces allow the model to trade inference-time compute for better performance (OpenAI, 2024; Yang et al., 2025a; Anthropic, 2025), but are typically treated as artifacts excluded from the message history with the exception of tool-use (DeepSeek-AI, 2026). Prior work has investigated monitoring these reasoning traces to preemptively detect when the model's final answer will be misaligned (Korbak et al., 2025; Bowman et al., 2022; Guan et al., 2025). Yet despite the ability of reasoning monitors to preemptively detect misaligned actions, research has found this approach has a blindspot where reasoning can become decoupled from the downstream answer (Turpin et al., 2023; Lanham et al., 2023; Howe & Carroll, 2026; Greenblatt et al., 2024).

Even with this decoupling blindspot, methods have been found for steering reasoning at inference time where the modified reasoning changes the final answer. Prior approaches either inject steering phrases into the reasoning trace at decoding time (Jin et al., 2025; Yang et al., 2025b; Zhang et al., 2025b; Wu et al., 2025; Muennighoff et al., 2025), or train the model to condition on special tokens inserted externally into its context (Goyal et al.; Kim et al., 2025; Wen et al., 2025). Some combine both, using special tokens the model is trained to respond to as steering phrases (Ringel et al., 2025). Other work has investigated training the model to produce these tokens during generation, however only in the context of the model's own self-correction (Zhang et al.; Sel et al., 2025). In each case, the signal acts on the reasoning rather than reporting on it. An overseer using any of these approaches must therefore still use the reasoning as a proxy for the final answer, which is precisely the mechanism that the decoupling undermines. We instead ask whether the proxy can be step-sided entirely.

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.

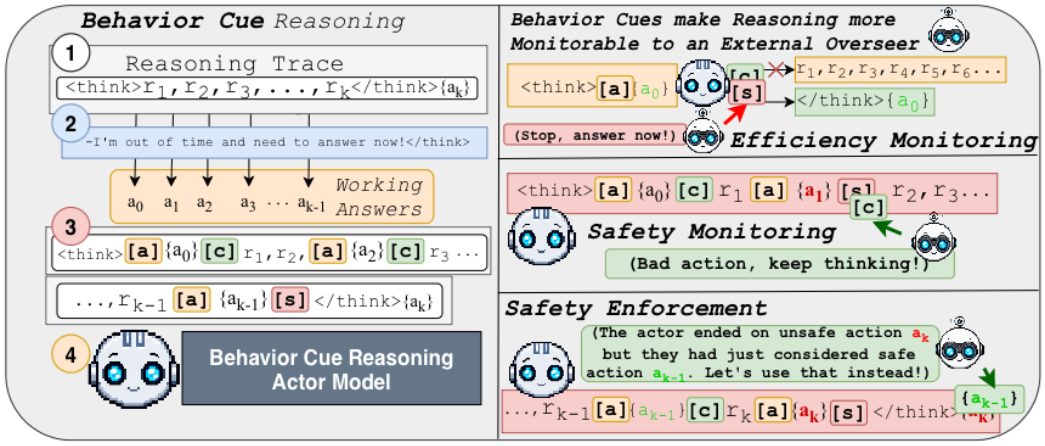

Figure 1: We take reasoning traces from a base actor model (1) and elicit the immediate working answer at each step by truncating the reasoning, appending a stop phrase, and allowing the model to produce an answer (2). Working answers are filtered to emulate the behavior of the working answer *only* surfacing when changing and embedded back into the reasoning trace with Behavior Cues (3). Post SFT (4), the Behavior Cue Reasoning Model adopts this answer reporting behavior, creating ideal decision points for an external oversight monitor. In **Efficiency Monitoring**, the monitor optimizes stopping once the correct final answer surfaces. In **Safety Monitoring**, the monitor attempts to prevent the commitment of unsafe actions by continuing reasoning. In **Safety Enforcement**, the monitor records the latest safe action to commit instead, allowing for secured unsafe reasoning.

We introduce Behavior Cues to progress scalable oversight by training the model to report on its own state by emitting special token sequences. The `[answer]` cue surfaces the model's current working answer, formatted for parsing, and appears only when that answer changes rather than at every reasoning step. The `[continue]` and `[stop]` cues follow each answer and mark whether the model is included to continue reasoning. We call this Behavior Cue Reasoning. As the current working answer is explicitly signaled within the reasoning, an external monitor can track answer progression through simple parsing, treating the model as a black box and removing the need to infer this answer from the reasoning. While signaling what the model would naturally do, the same cues can also be injected by an external monitor to induce the corresponding behavior, allowing for intervention in a model's reasoning and extending monitorability into controllability.

We evaluate on Qwen3-8B (Yang et al., 2025a) (hybrid reasoning) and GLM-Z1-9B (GLM et al., 2024) (pure reasoning) to test generalization across different model families in three separate problem domains. AIME serves as a single-turn evaluation for extended reasoning in complex math problem solving. Textworld, a situated text environment with natural language observations, evaluates Behavior Cues in an out-of-domain, multi-turn, task completion setting. Finally, we extend Hazardworld with a text-based interface to allow for evaluation of task completion under constraint adherence in multi-turn settings.

We evaluate Behavior Cues for scalable oversight through three research questions, one focused on the Behavior Cue Reasoning model itself and two on the monitorability of Behavior Cue Reasoning to external oversight. **RQ1:**To what degree does training a model to perform Behavior Cue Reasoning impact baseline task performance? **RQ2:**To what degree does a Behavior Cue Reasoning model adhere to externally enforced Behavior Cues? **RQ3:** To what degree does Behavior Cue Reasoning enable an external overseer to make a model safer and more efficient?

Our results demonstrate overall positive effects on baseline performance, adherence to external enforcement, and how Behavior Cues can enable oversight where standard reasoning would be otherwise noisy, opaque, or costly to monitor. We observe performance improvements in all domains and both model families. In the majority of cases, the actor models follow injected Behavior Cues with the expected behavior. Most significantly, Behavior Cues serve as a critical interface for external oversight. We find that by parsing candidate actions surfaced at `[answer]` against an oracle, a monitor is able to recover safe actions from 83% of reasoning traces that would otherwise propose

an unsafe action for Qwen3-8B and 68% for GLM-Z1-9B. Relative to the corresponding Behavior Cue Reasoning models without intervention, recovery raises average adjusted reward by 1.54 for Qwen3-8B and 1.62 for GLM-Z1-9B, using fewer tokens than the base-model retry baselines. We further show that the progression surfaced by `[answer]` can be equivalently used by early-stopping methods that would otherwise require repeated interruption and answer probes during reasoning. Through these experiments, we show that improved monitorability through Behavior Cue Reasoning enables oversight for both safety and efficiency.

In summary, our contributions are as follows:

- We introduce Behavior Cues, a mechanism for making LLM reasoning more monitorable and controllable for oversight through trained token sequences (`[answer]`,`[continue]`,`[stop]`) that act as surface-level hooks for reasoning behaviors.
- We show a model trained to perform Behavior Cue Reasoning displays overall equal to greater baseline performance and reliably adheres to expected behavior when the cue is enforced externally.
- We demonstrate that Behavior Cue Reasoning enables an oracle safety monitor to recover safe actions from up to 83% of otherwise unsafe reasoning traces, improving adjusted reward while using fewer tokens than the base-model retry baselines.
- We demonstrate that Behavior Cue Reasoning enables the deployment of early-stopping efficiency methods without the need for additional, repeated answer probes during generation.
- We publicly release our LLM-compatible Hazardworld extension as an artifact for reasoning oversight research.

2 Background

Reasoning traces. Formally, we define a reasoning trace as the content produced by an LLM before it commits to a final answer. For the models investigated in this work, these reasoning traces are enclosed within `<think></think>` tags. We define a trajectory step as the input-output pair corresponding to a single LLM call, and define a reasoning step as the portions of reasoning traces separated by two newlines. For example, the single-turn question-answer paradigm of AIME is a single trajectory step with potentially multiple reasoning steps. A trajectory is defined as the full sequence of trajectory steps made by the LLM when attempting a task. We define the working answer of an LLM as the answer the LLM would provide if its reasoning were terminated at any step.[1]

Scalable Oversight. Scalable oversight refers to the challenge of supervising AI systems whose skills or capabilities exceed that of evaluators (Amodei et al., 2016; Bowman et al., 2022). When evaluating the utility of Behavior Cue Reasoning in safety, we specifically use an oracle to separate the capability of the monitor to classify an answer as dangerous or not from the ability of monitor to extract intent from the reasoning. We discuss other works within the domain of scalable oversight in Section 6.

External Oversight. When evaluating the impact of Behavior Cues in making an LLM's reasoning more monitorable and controllable to oversight, we refer to the reasoning model as the *actor model*. While prior work has investigated internal model signals such as output distributions (Quamar & Areeb, 2025; Sharma & Chopra, 2025; Yang et al., 2025c) or hidden state representations (Azaria & Mitchell, 2023; Zhang et al., 2025a), we treat the actor model as a black box where only the generated tokens are observed. In Hazardworld, we evaluate the degree to a rule-based oracle can leverage the answer progression surfaced by `[answer]` to rescue generations that would have otherwise ended in an unsafe action by overwriting the final action with a safe action considered earlier. In AIME and Cookingworld, we examine how reliably previous work on black-box early-stopping methods such as ES-COT (Mao et al., 2025) and Dynasor (Fu et al., 2025). These methods were chosen because their stopping decisions depend only on the sequence of intermediate answers. Both obtain this sequence by repeatedly interrupting the model to elicit an answer, ES-CoT at every reasoning step and Dynasor at fixed token intervals. A Behavior Cue Reasoning model surfaces the same sequence through `[answer]`, so both stopping rules can be applied directly to its trace without any probing.

[1] In this work, we elicit the working answer by appending `- I'm out of time and need to answer!\n</think>`. However, we note in degenerate cases the LLM may not produce an end of sequence token after providing an answer.

3 BEHAVIOR CUES AND EXTERNAL OVERSIGHT

Behavior Cues. We define Behavior Cues as special token sequences an LLM is trained to naturally emit during reasoning immediately before specific behaviors. In this work, we explore the application of three Behavior Cues:

- `[answer]`: updates the working answer. We explicitly train the model to provide its best-guess answer without reasoning, resulting in a progression of answers throughout the reasoning phase.
- `[continue]`: after a new working answer, `[continue]` signals further reasoning token generation.
- `[stop]`: after a new working answer, `[stop]` signals a termination of reasoning.

Thus, when a model trained to produce these Behavior Cues is generating the answer to a question, `[answer]` should always appear at the start of reasoning immediately after the `<think>` tag and be followed by what is functionally the model's non-reasoning best guess answer. If the model then generates `[continue]`, it should proceed with its reasoning and produce another `[answer]` when its working answer changes. If the model generates `[stop]`, it should immediately stop its reasoning. These Behavior Cues thus are meant to serve a dual purpose of both signaling the occurrence of specific behaviors as well as acting as enforceable levers. Figure 1 shows how the actor model is trained to naturally produce Behavior Cues as well as how they can be used by an external module as a sample efficient interface for oversight. Attempts to elicit Behavior Cue Reasoning through prompting or RL were both unsuccessful. See Appendix B for more details.

Let $r_0, r_1, r_2, ..., r_{k-1}$ represent the steps in a model's reasoning and $a_0, a_1, a_2, ..., a_{k-1}$ represent the model's working answers for each corresponding reasoning step. We train a model to perform Behavior Cue Reasoning through an elicitation, embedding, and SFT pipeline. First we elicit the working answer at each reasoning step for every task in the training set by truncating reasoning, appending a termination phrase, and allowing the model to produce a final answer[2]. Then, these working answers are embedded into the reasoning between `[answer]` and `[continue]` (or `[stop]`) blocks to create the SFT dataset. At step r_n, the answer a_n is only embedded if it is not equal to the last embedded answer. This ensures each occurrence of `[answer]` is meaningful, marking a change in the model's most probable answer. To isolate the effects of any performance drift to training a model to perform Behavior Cue Reasoning specifically, we do not filter reasoning traces for correctness of the final answer. Training hyperparameters are documented in Appendix A.

External Oversight. We refer to the reasoning model being overseen as the *actor model* and treat it as a black box: the overseer observes only the tokens the actor generates. When evaluating the utility of Behavior Cue Reasoning for safety, we use an oracle that classifies an action as safe or unsafe according to the environment's ground-truth constraints. We use an oracle rather than a learned classifier, as the latter would confound our results with the classifier's own accuracy in judging the danger of an action. This allows us to isolate the benefits of Behavior Cue Reasoning. An oracle can only classify the actions it is given, and under standard reasoning the monitor must otherwise infer what action the actor model is currently considering. With Behavior Cue Reasoning, each candidate action surfaced by `[answer]` can be checked as reasoning proceeds without the need to infer intent. The overseer records the most recent safe candidate and substitutes it if reasoning ends on an unsafe action. If no safe candidate exists, it blocks the unsafe action, appends a warning as an observation, and re-queries the actor. Under standard reasoning, unsafe actions trigger the same warning and retry procedure.

For efficiency, the overseer applies the stopping rules of ES-CoT (Mao et al., 2025) and Dynasor (Fu et al., 2025) to the answer progression surfaced by `[answer]`, stopping reasoning once a rule is triggered. Applied to standard reasoning, the same rules require interrupting the actor to elicit an answer at a fixed or pre-defined interval. This answer probe effectively performs the same intent inference step-siding that motivates the `[answer]` cue. Therefore we use ES-CoT and Dynasor to evaluate the utility of Behavior Cue Reasoning for improving reasoning monitorability in the efficiency setting. When applying these methods, we do not use the same answer probe at interval. Rather, we treat the last answer a_n surfaced by `[answer]` as the answer with which the model would respond at any point with an answer probe between this answer and the next `[answer]` cue.

[2]This approach is similar to what is done by Mao et al. (Mao et al., 2025)

4 EXPERIMENTAL SET UP

We evaluate Behavior Cues across two model families (Qwen3-8B and GLM-Z1-9B), and three domains, (AIME, Textworld, and Hazardworld). We use Qwen3 as a hybrid reasoning model, trained to both reason and immediately answer, and GLM as a pure reasoning model to evaluate the transferability of our method across architectures and training regimes. We use AIME, Textworld, and Hazardworld to evaluate Behavior Cues in making an actor model's reasoning trace more monitorable and controllable to oversight across a diverse set of domains. Our experiments investigate each stage at which Behavior Cues take effect from the base model itself to their impact through an external oversight monitor. For additional details, see Appendix Section C for Cookingworld, and Section D for Hazardworld.

AIME: The AIME (American Invitational Math Examination) problem set is a series of complex math problems. We use the 2025 AIME question set as an evaluation split to minimize potential data contamination with the pre-2025 AIME question sets acting as our training set for SFT. For AIME reasoning traces, the monitor has two objectives. For successful reasoning traces, the monitor is tasked with stopping reasoning as soon as the correct answer first surfaces. For dead-end reasoning traces, the monitor is tasked with stopping reasoning as soon as possible.

Cookingworld: Textworld Côté et al. (2018) is a situated text simulator originally designed for training classical RL agents in natural language environments. For our experiments, we use the Cookingworld scenarios used for evaluation in Cui et al. (2025) with alternate seeds as our train split. Cookingworld tasks involve the LLM acting as an agent to navigate a household setting, gather ingredients, and prepare a meal. As ingredients are limited, failure can occur if an important ingredient is consumed or prepared in the wrong way such as eating a raw ingredient that should have been cooked. If this occurs, the environment immediately terminates with a losing message. The specific tasks and environment layouts used in Cui et al. (2025) are used as our test split with alternate recipes and environment layouts used as our train split. Please see Appendix C for details.

Hazardworld: Hazardworld Yang et al. (2021) is a Minigrid environment originally designed for training RL agents. In Hazardworld, the agent must navigate to a series of objects while avoiding a set of randomly generated hazard tiles. The specific type of hazard tile the agent must avoid is provided in a natural language description such as `Avoid Water` or `Stay at least 1 block away from Lava`. Unsafe actions in Hazardworld violate the provided constraints. We report average adjusted reward: each episode's total reward is set to zero if its constraint violation budget is exceeded, then averaged across episodes. For our experiments, we extend Hazardworld to be compatible with LLM agents. See Appendix D for a comprehensive list of modifications to the original environment. In Hazardworld, the monitor is tasked solely with the classification of actions as safe or unsafe given the current actor model context.

5 EVALUATION

We center our evaluation around three research questions investigating the actor model and oversight module. These questions evaluate the actor model across downstream performance and controllability, and the Behavior Cue Reasoning traces themselves in how they lend themselves to better enabling an external oversight module to perform efficiency monitoring and safety monitoring.

RQ1: To what degree does training a model to perform Behavior Cue Reasoning impact baseline task performance? Methods for improving the controllability or monitorability of models typically come at the cost of some negative impact on performance. In these initial experiments, we examine the degree of performance drift from fine-tuning an actor model to perform Behavior Cue Reasoning as well as how controllable the actor model is to externally enforced Behavior Cues. To evaluate task performance, we fix decoding parameters per domain and report performance on a held-out test set. We examine only the actor model in isolation to determine what the cost of Behavior Cues is in the oversight-performance trade-off.

Training a model to perform Behavior Cue Reasoning has minimal to positive effects on baseline task performance. Table 1 shows the Behavior Cue Reasoning model matches or surpasses base model performance across all domains. We observe this to be in part a baseline benefit of Behavior Cues in rescuing reasoning traces that hit the context cap and otherwise terminate without an answer.

Table 1: Top: task performance. Bottom: adherence to forced reasoning cues (continue / stop), for the Behavior Cues variants. The Hazardworld **adjusted score** is calculated on the basis of a 0 overall reward for trajectories where the constraint violation budget was exceeded. See Table 2 for a more detailed breakdown. We see overall performance gains across the board for all models.

Model	**AIME2025** (% Correct)	**Cookingworld** (% Wins)	**Hazardworld** (Adj. Score)
Qwen3-8B	62.5	53.3	2.3
Qwen3-8B w/ Behavior Cues	69.2	56.7	2.4
GLM-Z1-9B	54.4	36.6	1.3
GLM-Z1-9B w/ Behavior Cues	56.7	46.7	1.3
	Adherence to forced reasoning cues (% continue / % stop)		
Qwen3-8B	30.9 / 100.0	84.0 / 55.0	97.0 / 100.0
GLM-Z1-9B	34.4 / 100.0	75.5 / 63.5	87.0 / 98.0

As the model always outputs an initial answer without reasoning immediately after the `<think>` sequence, the most recent answer surfaced by `[answer]` acts as a fallback. We find this recovers roughly 10% of turns for the Behavior Cue Reasoning model.

RQ2: To what degree does a Behavior Cue Reasoning model adhere to externally enforced Behavior Cues? To evaluate Behavior Cues controllability, we randomly sample 200 decision points for each Behavior Cue Reasoning model where the model would output `[continue]` or `[stop]` (100 each), replace it with the opposite reasoning cue, and allow the model to continue generating. We measure the rate of the actor model adhering to the replaced cue. Adherence to `[continue]` is calculated as the number of times the model generates at least one or more `[answer]` blocks before terminating reasoning. Adherence to `[stop]` is calculated as the number of times the model immediately generates the `</think>` tag and reports the last reported answer. While adherence to `[stop]` could be deterministically implemented by enforcing the `EOS` token instead of `[stop]`, we report proper end-think behavior instead for scientific rigor. In practice, both the proper end-think behavior elicited by `[stop]` and forcible termination by `EOS` are functionally identical as the last reported working answer is still present in the partial reasoning trace in proper format.

Adherence to externally enforced Behavior Cues for a Behavior Cue Reasoning model differs by domain. The bottom half of Table 1 shows the adherence to externally enforced `[continue]` and `[stop]`. While we see overall adherence, there is a clear stratification by domain. In Hazardworld, reasoning traces can be long (up to 8k tokens) but the model sees many examples of `[proposed action]` followed by `[stop]` due to the smaller action space. In contrast, reasoning traces from Cookingworld are overall far shorter (up to 2k tokens) and typically involve a similar set of actions. As a result, the model learns the `[stop]` behavior to be associated with a specific subset of actions and reasoning length. In particular, we note adherence to `[stop]` has an inverse relationship to the reasoning length with practically zero adherence when injected at the very first `[answer]` block, rapidly increasing as reasoning continues and switches between more potential actions. AIME shows the other extreme with an unbounded answer space and highly variable length reasoning traces. Only `[stop]` is learned with the specific behavior we intended. In AIME specifically, we note that once a high-confidence answer is surfaced, the model never deviates, and reasoning quickly terminates again on the same answer if forced to continue. Notably, we observe **no answer changes** even when Qwen adheres to `[continue]`, and only a 16% answer-change rate for GLM. Across all Behavior Cue Reasoning traces for both models, we observe a **less than 3% rate** of the model proposing the correct answer only to later change to another answer.

RQ3: To what degree does Behavior Cue Reasoning enable an external overseer to make a model safer and more efficient? We evaluate the utility afforded by Behavior Cue Reasoning for an external overseer through two domains, safety and efficiency. In Hazardworld, we test both safety and efficiency by comparing the adjusted reward and total tokens used by comparing a retry baseline against the recovery enabled by Behavior Cues. In the retry setting, if the model attempts to perform an unsafe action, a warning is instead appended as an observation, the action is blocked from the

environment, and the model is queried again. In the recovery setting, if the Behavior Cue Reasoning model concludes with an unsafe action, any earlier surfaced safe action instead overrides the final action. If no safe action was proposed, the unsafe action is blocked and the model is queried again with a warning appended as an observation. Safe and unsafe actions are directly queried against the environment engine. For AIME and Cookingworld, we apply ES-CoT and Dynasor to evaluate if similar savings are achieved on the Behavior Cue Reasoning models as base models without the need for answer probes. See Appendix E for the precise ES-Cot and Dynasor settings used per domain.

Table 2: Adjusted reward is the average episode reward after setting each episode's total reward to zero if constraint violations exceed its budget. No intervention are runs with no monitor. Recovery tests the performance of Behavior Cue Reasoning models when an oracle can select previously considered safe actions when reasoning concludes on an unsafe action. Retry re-queries the model when the action is unsafe with a warning message. Safe and unsafe actions are queried directly from the environment. In the recovery setting, the Behavior Cue Reasoning model both scores higher and uses far less tokens than the retry setting.

Configuration	Adj. Reward	Avg. Reward	Gen. Tokens
No Intervention			
Qwen3-8B	2.35	4.86	3.6M
Qwen3-8B w/ Behavior Cues	2.45	4.64	3.1M
GLM-Z1-9B	1.25	3.73	5.7M
GLM-Z1-9B w/ Behavior Cues	1.27	3.80	4.9M
Recovery			
Qwen3-8B + Retry	3.40	3.40	4.6M
Qwen3-8B w/ Behavior Cues + Recovery	**3.99**	3.99	3.2M
GLM-Z1-9B + Retry	2.67	2.67	8.3M
GLM-Z1-9B w/ Behavior Cues + Recovery	**2.89**	2.89	4.3M

Behavior Cue Reasoning enables recovery of safe actions from reasoning that ultimately ends unsafely, improving adjusted reward with fewer tokens than the retry baseline. The oracle monitor recovers safe actions from 83% and 68% of otherwise unsafe reasoning traces for Qwen3-8B and GLM-Z1-9B, respectively. Table 2 shows that, relative to the corresponding Behavior Cue Reasoning models without intervention, oracle recovery increases average adjusted reward from 2.45 to 3.99 for Qwen3-8B (+1.54, approximately 63%) and more than doubles it for GLM-Z1-9B, from 1.27 to 2.89 (+1.62). While the Retry baselines also find improved scores, this improvement comes at a major token cost due to the need to repeatedly re-query the model upon the rejection of unsafe actions, requiring 27% more tokens for Qwen3-8B and 45% more tokens for GLM-Z1-9B. In comparison, Behavior Cue Reasoning combines both safety and efficiency, allowing for a greater performance gain at roughly the same token cost as the base model.

Behavior Cue Reasoning can be used by black-box methods that otherwise require answer probes. Table 3 compares performance of ES-CoT and Dynasor on the base model with answer probes versus on Behavior Cue Reasoning models where the probes for any point instead rely on the most recent answer surfaced by `[answer]`. In AIME, Dynasor on Behavior Cue Reasoning Qwen3-8B exhibits a similar accuracy–efficiency trade-off to the base model: 13.5% versus 11.8% of generated tokens saved, with accuracy reductions of 3.3 versus 1.1 percentage points relative to each model without early stopping. Probe generation substantially erodes the savings of the base models: once probe tokens are counted, savings drop to 5.0% for Qwen3-8B Dynasor and 1.5% for GLM-Z1-9B Dynasor, and base Qwen3-8B ES-CoT consumes more tokens than it saves (-9.3%). The effect is more pronounced in Cookingworld, where short reasoning traces leave little room for probe overhead to be amortized: base GLM-Z1-9B Dynasor saves 4.3% of trajectory tokens but costs 12.3% more once probes are included, whereas carrying forward the surfaced `[answer]` saves 3.1% with no loss in agreement. ES-CoT is effectively unusable in Cookingworld for all models due to the shorter length of the reasoning.

Table 3: Early stopping with ES-CoT and Dynasor. ΔAcc is the accuracy change (percentage points) relative to the same model without early stopping. Saved is the percentage of generated tokens removed from the reasoning trajectory; w/ probes additionally charges the tokens generated by the answer probes base models need to obtain intermediate answers. Behavior Cues models read the latest surfaced `[answer]` and require no probes. In Cookingworld, accuracy is agreement with the action the full reasoning trace commits to.

	Qwen3-8B					GLM-Z1-9B				
	Base			Behavior Cues		Base			Behavior Cues	
Method	ΔAcc	Saved	w/ probes	ΔAcc	Saved	ΔAcc	Saved	w/ probes	ΔAcc	Saved
AIME2025										
ES-CoT	−5.6	5.2%	−9.3%	−10.0	8.6%	+1.1	12.3%	8.4%	−5.6	12.6%
Dynasor	−1.1	11.8%	5.0%	−3.3	13.5%	+1.1	8.0%	1.5%	−3.3	9.0%
Cookingworld										
ES-CoT	0.0	0.6%	−1.0%	0.0	0.0%	0.0	0.1%	−1.4%	0.0	0.3%
Dynasor	−1.1	8.1%	4.4%	−6.4	8.8%	+0.6	4.3%	−12.3%	+1.1	3.1%

6 Related Works

Reasoning Monitorability and Oversight: Recent work examines the potential of monitoring LLM reasoning to detect and address misaligned behaviors before they surface in the final response (Korbak et al., 2025; Bowman et al., 2022; Guan et al., 2025). Previous work has shown that these reasoning traces can still be unfaithful, where the final answer of the model doesn't align with the reasoning (Turpin et al., 2023; Lanham et al., 2023; Howe & Carroll, 2026; Greenblatt et al., 2024). This unfaithfulness becomes an issue when an overseer needs to infer the actor model's working answer from the reasoning, as unfaithfulness may reduce the monitor's precision in identifying the actor's current working answer. Baker et al.(Baker et al., 2025) demonstrate that while reasoning monitors can detect misaligned behaviors in frontier reasoning models, applying optimization pressure directly to the reasoning trace can produce obfuscated misbehavior, where the model learns to hide misaligned behavior rather than avoid them to improve reward: this motivates a "monitorability tax", where reasoning is deliberately left outside the training loop to explicitly isolate it from optimization pressures. Behavior Cues aligns with this principle and side steps the issue of faithfulness by training an actor model to directly expose its working answer at meaningful junctions. All optimization pressure for oversight is instead placed on the monitor and interpreting the actor model's reasoning becomes a non-issue.

Reasoning Control through Token Injection: Prior work examines the application of interventions for steering a model's reasoning. These approaches typically involve the injection of natural language phrases to steer the model towards certain thought paths without modification of the underlying model Jin et al. (2025); Yang et al. (2025b); Zhang et al. (2025b); Wu et al. (2025); Tang et al. (2025). (Muennighoff et al., 2025) train a non-thinking model to reason effectively, showing performance can be further improved by early termination or the use of the phrase "Wait" to extend reasoning. (Zhao et al., 2025) leverage activations and the "wait" token to elicit long chain of thought behavior with minimal training. In contrast to these methods, our approach specifically targets making a model's reasoning trace more monitorable **and** steerable.

Inducing Behavior Through Control Tokens: Previous works have explored the use of controlling model behavior via special tokens, both inside and outside of a reasoning trace. (Zhang et al.) leverage the [RESET] token to allow a model to completely undo potentially unsafe generations with (Sel et al., 2025) refining this to a specific subset of the generation. More recent work extend the use of special tokens to reset behavior to reasoning (Yang et al., 2025d). (Ringel et al., 2025) demonstrate how a specialized token for enforcing continued reasoning can outperform a natural language signal such as "Wait". (Kim et al., 2025) and (Goyal et al.) use the [PAUSE] token albeit in different ways, the former allows the model to express low confidence and the latter to signal the model to perform more internal hidden layer calculations, effectively increasing the compute used for generating the next token.

Early Stopping in Reasoning: In this section, we discuss work that explicitly terminate or truncate reasoning early for both better token efficiency and performance. These approaches typically rely on entropy, or the confidence in a proposed answer as a signal to immediately terminate the model's reasoning at a certain reasoning step (Yang et al., 2025c; Sharma & Chopra, 2025; Quamar & Areeb, 2025). (Sun et al., 2025) uses the similarity of newly generated reasoning steps with previous steps as a signal for early termination. (Mao et al., 2025) generates answers for each reasoning step, ending reasoning when the answer is consistent through a large number of steps.

Reasoning Agents in Interactive Tasks Agents in interactive environments have been studied long before the advent of LLMs with language-focused testbeds such as Textworld Côté et al. (2018) or Jericho Hausknecht et al. (2020) used for training RL agents in sequential decision-making Shridhar et al.; Wang et al. (2022). These agents often used a set of action templates to bound natural language space to a subset learnable by a smaller network. Recent work has examined the performance of LLMs as the decision making centers of agents in these environments Cui et al. (2025); Paglieri et al. (2024). LLMs as agents in interactive tasks have also been explored in a wide range of domains such as coding Yuan et al. (2025); Yang et al. (2024), scientific discovery Nathani et al. (2025); Jansen et al. (2024), and deep research Huang et al. (2025).

7 Considerations

Limitations. Our work progresses scalable oversight through the explicit training of a model's reasoning to be more monitorable and controllable to external intervention. While we attempt to minimize downstream performance impact, there is a clear distribution shift. Regardless of whether this ultimately improves performance, future work may investigate methods for teaching a model to perform Behavior Cue Reasoning without needing to modify the original model's weights at all. We primarily investigate domains where the final answer is a short phrase that can easily be integrated into the reasoning trace. Another direction may be pushing the model to specifically record this answer in an external document to facilitate extension to environments where the downstream answer may be longer than the reasoning itself. Lastly, we primarily investigate the trio of `[answer]`, `[continue]`, and `[stop]` due to the implicit relation between all of these behaviors. Future work may investigate other reasoning behaviors such as backtracking for verification.

Societal Impacts. Behavior Cues are intended to improve scalable oversight by making intermediate reasoning states easier to monitor and intervene on, potentially reducing wasted inference compute and preventing unsafe commitments before they appear in final outputs. However, because many users interact with LLMs through APIs or mediated interfaces, cue-based control could be hidden from end users and used by a platform or intermediary to silently steer model reasoning without meaningful transparency or consent. Behavior Cues should also be understood as strong learned priors rather than deterministic guarantees due the stochasticity of LLM generation, reflected by the varying degrees of cue compliance observed across domains. See Appendix H for further discussion.

8 Conclusion

We introduced Behavior Cues, a method for making LLM reasoning more monitorable and controllable through explicit intermediate answers and cues for controlling reasoning. Across two model families and three domains, fine-tuning for Behavior Cue Reasoning maintained or improved baseline task performance. Models also responded to externally enforced `[continue]` and `[stop]` cues, although compliance varied by domain and cue type.

The answer progression exposed by `[answer]` supports external oversight for both safety and efficiency. In Hazardworld, an oracle monitor recovers previously considered safe actions when reasoning ends on an unsafe action, improving adjusted reward while using fewer generated tokens than the base-model retry baseline. In AIME and Cookingworld, ES-CoT and Dynasor can use these surfaced answers without additional answer probes, avoiding probe-generation overhead while exhibiting accuracy–efficiency trade-offs. These results suggest that training models to expose intermediate decisions can make reasoning easier to oversee. Behavior Cues do not guarantee safe or correct behavior: their utility depends on cue compliance and the capabilities of the overseer. Extending this interface to broader tasks and evaluating it with learned safety monitors are directions for future work.

AI USE STATEMENT

Generative AI tools were used to provide feedback on manuscript drafts and to assist with running experiments. The initial codebase was developed without AI assistance; AI tools were subsequently used to refine the codebase and support code cleanup and anonymization for submission. AI tools were also used to check the manuscript's compliance with ICLR formatting requirements. The authors are responsible for the final manuscript, code, and experimental results.

REPRODUCIBILITY STATEMENT

We provide code for running all experiments reported in this paper. Numerical instability may introduce variation in the results, so the reported values may not be reproduced exactly.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Anthropic. Building with extended thinking. https://docs.claude.com/en/docs/build-with-claude/extended-thinking, 2025.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pp. 41–75. Springer, 2018.

Christopher Zhang Cui, Xingdi Yuan, Ziang Xiao, Prithviraj Ammanabrolu, and Marc-Alexandre Côté. Tales: Text adventure learning environment suite, 2025. URL https://arxiv.org/abs/2504.14128.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling llm reasoning with certaindex, 2025. URL https://arxiv.org/abs/2412.20993.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*.

Ryan Greenblatt, Carson Denison, Benjamin Wright, Fabien Roger, Monte MacDiarmid, Sam Marks, Johannes Treutlein, Tim Belonax, Jack Chen, David Duvenaud, Akbir Khan, Julian Michael, Sören Mindermann, Ethan Perez, Linda Petrini, Jonathan Uesato, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, and Evan Hubinger. Alignment faking in large language models, 2024. URL https://arxiv.org/abs/2412.14093.

Melody Y. Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y. Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability, 2025. URL https://arxiv.org/abs/2512.18311.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7903–7910, 2020.

David Herel and Tomas Mikolov. Collapse of self-trained language models, 2024. URL https://arxiv.org/abs/2404.02305.

Nikolaus Howe and Micah Carroll. The ends justify the thoughts: Rl-induced motivated reasoning in llm cots, 2026. URL https://arxiv.org/abs/2510.17057.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, et al. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096*, 2025.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. *Advances in Neural Information Processing Systems*, 37:10088–10116, 2024.

Hyunbin Jin, Je Won Yeom, Seunghyun Bae, and Taesup Kim. "well, keep thinking": Enhancing LLM reasoning with adaptive injection decoding. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9989–10018, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.520. URL https://aclanthology.org/2025.findings-acl.520/.

Eunki Kim, Sangryul Kim, and James Thorne. Learning to insert [pause] tokens for better reasoning. *arXiv preprint arXiv:2506.03616*, 2025.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, Scott Emmons, Owain Evans, David Farhi, Ryan Greenblatt, Dan Hendrycks, Marius Hobbhahn, Evan Hubinger, Geoffrey Irving, Erik Jenner, Daniel Kokotajlo, Victoria Krakovna, Shane Legg, David Lindner, David Luan, Aleksander Mądry, Julian Michael, Neel Nanda, Dave Orr, Jakub Pachocki, Ethan Perez, Mary Phuong, Fabien Roger, Joshua Saxe, Buck Shlegeris, Martín Soto, Eric Steinberger, Jasmine Wang, Wojciech Zaremba, Bowen Baker, Rohin Shah, and Vlad Mikulik. Chain of thought monitorability: A new and fragile opportunity for ai safety, 2025. URL https://arxiv.org/abs/2507.11473.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models, 2025. URL https://arxiv.org/abs/2509.14004.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, 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/.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

Mohammad Atif Quamar and Mohammad Areeb. Logit-entropy adaptive stopping heuristic for efficient chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2511.04654.

Liran Ringel, Elad Tolochinsky, and Yaniv Romano. Learning a continue-thinking token for enhanced test-time scaling. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 3324–3345, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. URL https://aclanthology.org/2025.ijcnlp-long.177/.

Bilgehan Sel, Dingcheng Li, Phillip Wallis, Vaishakh Keshava, Ming Jin, and Siddhartha Reddy Jonnalagadda. Backtracking for safety. *arXiv preprint arXiv:2503.08919*, 2025.

Aman Sharma and Paras Chopra. Think just enough: Sequence-level entropy as a confidence signal for llm reasoning, 2025. URL https://arxiv.org/abs/2510.08146.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2510.10103.

Siao Tang, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Concisehint: Boosting efficient reasoning via continuous concise hints during generation. *arXiv preprint arXiv:2506.18810*, 2025.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Budgetthinker: Empowering budget-aware llm reasoning with control tokens. *arXiv preprint arXiv:2508.17196*, 2025.

Tong Wu, Chong Xiang, Jiachen T Wang, G Edward Suh, and Prateek Mittal. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*, 2025.

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

Chenxu Yang, Qingyi Si, Mz Dai, Dingyu Yao, Mingyu Zheng, Minghui Chen, Zheng Lin, and Weiping Wang. Test-time prompt intervention, 2025b. URL https://arxiv.org/abs/2508.02511.

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*, 2025c.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: agent-computer interfaces enable automated software engineering. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 50528–50652, 2024.

Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34:13794–13808, 2021.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*, 2025d.

Xingdi Yuan, Morgane M Moss, Charbel El Feghali, Chinmay Singh, Darya Moldavskaya, Drew MacPhee, Lucas Caccia, Matheus Pereira, Minseon Kim, Alessandro Sordoni, and Marc-Alexandre Côté. debug-gym: A text-based environment for interactive debugging, 2025. URL https://arxiv.org/abs/2503.21557.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025a.

Xingsheng Zhang, Luxi Xing, Chen Zhang, Yanbing Liu, Yifan Deng, Yunpeng Li, Yue Hu, and Chenxu Niu. Can we steer reasoning direction by thinking intervention? In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3888–3913, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.209. URL https://aclanthology.org/2025.findings-emnlp.209/.

Yiming Zhang, Jianfeng Chi, Hailey Nguyen, Kartikeya Upasani, Daniel M Bikel, Jason E Weston, and Eric Michael Smith. Backtracking improves generation safety. In *The Thirteenth International Conference on Learning Representations*.

Zekai Zhao, Qi Liu, Kun Zhou, Zihan Liu, Yifei Shao, Zhiting Hu, and Biwei Huang. Activation control for efficiently eliciting long chain-of-thought ability of language models. *arXiv preprint arXiv:2505.17697*, 2025.

A Behavior Cue SFT Hyperparameters

We full-parameter fine-tune Qwen3-8B with TRL using bfloat16 precision, SDPA, and completion-only loss. We use per-device training and evaluation batch sizes of 2 with no gradient accumulation, and train for 5000 updates at a learning rate of 5×10^{-5} with a linear schedule and no warmup. We set the maximum sequence length to the longest tokenized training or validation example plus $2{,}048$ tokens. We use the same training settings for GLM-Z1-9B.

B Embedded Answer Fixation

When initially attempting to train models to perform Behavior Cue Reasoning, our initial approaches were pure prompting and RL fine-tuning. However, we found both to fail due to the models' tendency to fixate on any previously mentioned answers within their reasoning trace. We found models required

SFT to properly adhere to the format of Behavior Cue Reasoning where we observed an upside-down U curve in terms of performance, where early checkpoints perform strictly worse than the base model while failing to adhere to the Behavior Cue Reasoning format; middle checkpoints both achieved proper formatting and comparable, if not greater, performance to the base model; and later checkpoints once again demonstrating a degradation in base model capabilities even while adhering to the Behavior Cue Reasoning format.

We suspect the early collapse to be a symptom of the model initially fixating on the embedded answers during early training, resulting in a collapse of reasoning capabilities. The middle checkpoints see a sharp increase in performance as the model learns to adhere to the Behavior Cue Reasoning format without disrupting its natural reasoning. The latter checkpoints collapse again as the model effectively begins to train on its own reasoning output. The later checkpoint collapse is a studied effect that has evidence tracing back to GPT-2 (Herel & Mikolov, 2024). For the early checkpoint result and question of why Behavior Cues cannot simply be elicited by SFT or RL, we perform a small scale experiment.

This experiment is performed post-hoc with a Behavior Cue Reasoning model to prove that a base model will otherwise fixate to any answers included in its context rather than surface them and continue reasoning as done in Behavior Cue Reasoning. We take 100 context and reasoning trace pairs, generated by a Behavior Cue Reasoning model. For the reasoning, we truncate to the second `[answer]` block, feed the context and truncated reasoning back to a non-Behavior Cue Reasoning model, and allow it to continue generating. This advantages the base model towards generating a different answer than that exposed in the first `[answer]` block, as the stretch of reasoning after the first answer that leads the Behavior Cue Reasoning to propse the second answer is explicitly provided to the base model. Despite this, we found the base model would fixate and repeat the first answer **73%** of the time.

During RL training, when the base model needs to instead construct a line of reasoning that leads it to a different answer itself, we saw this initial answer fixation almost always resulted in the second answer being a repeat of the first, resulting in non-convergence during training.

A second motivation for avoiding RL training arises from the "monitorability tax" coined by Baker et al(Baker et al., 2025). RL training a model to perform Behavior Cue Reasoning would explicitly require monitoring the reasoning trace to assign reward to ensure that the correct behavior is adhered to. This both creates a dual optimization pressure, as the training objective would also require some pressure to ensure the model actually arrives at the correct final answer, as well as allowing for the potential of reward hacking.

C COOKINGWORLD DATA FILTERING

Cookingworld tasks involve the LLM acting as an agent to navigate a household setting, gather ingredients, and prepare a meal. As ingredients are limited, failure can occur if an important ingredient is consumed or prepared in the wrong way, for example, eating a raw ingredient that should have been cooked. If this occurs, the environment immediately terminates with a losing message. The specific tasks and environment layouts used in Cui et al. (2025) are used as our test split with alternate recipes and environment layouts used as our train split. A max step count of 20 is used as a tractable horizon that still gives the LLM sufficient attempts to make non-terminal mistakes in the environment. A reasoning budget of 2048 tokens per step is allocated as higher token budgets resulted in minimal increases in performance for the base actor model. Due the difficulty of the environment, both models failed to complete almost all of the included scenarios with a minimal prompt. Following advice from Cui et al. (2025), we include the contents of the "`help`" in the initial observation to aid the models in completing the given tasks. We do not use the reasoning dead-end formulation in Cookingworld as due to the multi-step nature of the task, there is no ground-truth for if an action is 'incorrect' in the immediate context as it may lead to success or failure later on. While task failure is possible in Cookingworld, for example eating a required ingredient for the meal when there are no backups, labeling all reasoning for these trajectories with the reasoning-dead-end formulation would result in a number of reasoning traces over identical or near-identical states being labeled with the reasoning dead-end labels as well as the successful reasoning trace labels, resulting in a significant amount of noise. Furthermore, as the failure turns make up less than 5% of the total data samples, we discard them to focus monitor evaluation in Cookingworld on efficiency in a multi-turn setting.

D HAZARDWORLD CHANGES

We build on the HazardWorld benchmark of Yang et al. (Yang et al., 2021), with modifications grouped into four categories: task-tractability adjustments to make the environment solvable by LLM agents, a text observation and action interface, the scenario suite used for train/test evaluation, and benchmark-validity corrections to the original implementation.

D.1 TASK-TRACTABILITY ADJUSTMENTS

The original HazardWorld places its three pickup objects (Ball, Box, Key) and hazard tiles independently. On large grids this frequently produces scenarios in which an object is unreachable within a useful step budget, or in which the constraint never interacts with the optimal trajectory. To make episodes both tractable for LLM agents and informative about constraint adherence, we add two placement mechanisms and expose hazard density as a hyperparameter.

Chain object placement. When a ground-truth step budget `max_steps_gt` is supplied, objects are placed sequentially along a chain Agent $\rightarrow$ Ball $\rightarrow$ Box $\rightarrow$ Key, with each leg constrained to lie between 2 and $\lfloor \texttt{max_steps_gt}/4 \rfloor$ Manhattan cells from the previous waypoint. The lower bound of 2 guarantees at least one intermediate cell on each shortest path (used by hazard injection below); the upper bound bounds the optimal trajectory length.

Hazard-on-path injection. After object placement, we ensure that at least one tile of the avoid-hazard type lies within the bounding box of every consecutive (waypoint, waypoint) pair. If no such tile already exists, we place one on an empty cell, or otherwise overwrite a non-avoid hazard. This guarantees that the agent must reason about the constraint in order to traverse the chain. For `HazardWorldSequential` with `isBefore=False`, we use the dormant avoid object (the active `avoid_obj` is `None` until the trigger fires).

Hazard density. We expose hazard density as a hyperparameter `sparsity`, with default 0.75 (raised from 0.50 in the original). At 0.75, $\sim 61\%$ of interior cells are hazardous and $\sim 21\%$ are of the avoid type, increasing the rate of constraint engagement per episode. We also reorder placement in the Budgetary and Sequential subclasses to agent $\rightarrow$ objects $\rightarrow$ hazards (matching Relational), so that chain placement always operates on an empty interior.

Parameter	Default	Effect
`size`	13	Grid side length (now passable to all subclasses)
`max_steps_gt`	None	Ground-truth step budget; enables chain placement
`sparsity`	0.75	Probability each interior cell is a hazard

Table 4: New environment parameters added to `HazardWorldBase`.

D.2 TEXT OBSERVATION AND ACTION INTERFACE

We wrap the environment with a `HazardWorldTextWrapper` that produces ASCII observations and accepts text actions, enabling LLM agents to play without bespoke tokenisation.

Grid encoding. Every cell is rendered as a fixed two-character code, chosen so that each code is a single token in the Qwen3-8B tokenizer used in our experiments: `##` Wall, `..` Empty, `Lv` Lava, `Gr` Grass, `Wa` Water, `Bl` Ball, `Bo` Box, `Ky` Key, `??` Unseen. The agent is rendered as »/«/`vv`/`^^` in facing mode, where the agent can only move forward and must explicitly turn to change direction, or `Ag` in cardinal mode. Single-token encoding avoids cross-cell token boundaries that would otherwise force the model to spend reasoning capacity on parsing the map.

Observation and action options.

- *Partial observability:* a configurable square window (default 3×3) around the agent; cells outside the window are rendered `??`. For window size w the visibility radius is $r = (w-1)/2$.
- *Fog of war* (optional): once-seen cells remain revealed for the rest of the episode.

- *Cardinal movement* (optional): the four-action set {`up`, `down`, `left`, `right`} replaces the original turn-then-forward action set.
- *Auto-pickup:* moving onto a pickupable object collects it automatically, matching the convention common in text-game baselines.
- *Action parsing:* actions are extracted from `<action>·</action>` tags in model output.

D.3 SCENARIO SUITE

We release a fixed scenario file (`scenarios.json`) containing 30 training and 30 testing scenarios. Both splits share the same parameter combinations but use disjoint random seeds, so test-time evaluation measures generalisation to unseen layouts under matched constraints. Coverage is:

- **Budgetary** (10): three avoid-hazard objects $\times$ hazard counts $\texttt{hc} \in \{1, 3, 5, 6\}$.
- **Sequential** (10): five with `isBefore=true` and five with `isBefore=false`, varying the (`first_obj`, `avoid_obj`) pair.
- **Relational** (10): three avoid-hazard objects $\times$ $\texttt{min_dist} \in \{1, 2\}$. The choice to omit $\texttt{min_dist} \in \{0, 3\}$ is justified in §D.4.

D.4 BENCHMARK-VALIDITY CORRECTIONS

While porting HazardWorld we identified three pre-existing issues that affect the validity of comparisons drawn from the original implementation. We document and correct them here.

Non-deterministic scenario generation. The original `_gen_grid` mixes Gymnasium's seeded `self.np_random` (used for agent and object placement) with Python's unseeded `random` module (used for constraint parameters and hazard placement) and an unseeded `np.random.choice` call in `make_budgetary_constraint`. Consequently, a fixed environment seed does not reproduce the same scenario across runs. We apply standardization so that seed reproducibility is ensured.

Sequential constraint never deactivates. `HazardWorldSequential.step()` contains two bugs on the `isBefore` branch in the original repository:

- `curr_cell == first_obj` compares an object instance against an undefined local; the intended check is `curr_cell.type == self.first_obj`.
- `self.avoid_obj == None` is a comparison rather than an assignment; the active constraint is therefore never deactivated after the trigger fires.

Both are corrected. Because the second bug silently inverts the semantics of `isBefore=true` scenarios, prior numerical results obtained on the original implementation in this regime should be interpreted with caution.

Relational `min_dist` is not enforced. The relational constraint nominally requires the agent to remain at least `min_dist` Manhattan-cells from the avoid object. However, the original `step()` only registers a violation when the agent steps *onto* the avoid tile (`curr_cell.type == self.avoid_obj`), never when it merely enters the surrounding disk. Two consequences follow:

- `min_dist=0` is trivially always satisfied and therefore semantically empty.
- `min_dist=2` and `min_dist=1` produce identical runtime behavior despite being intended as different difficulty levels.

Additionally, under the partial-observation window of size 5 (radius $r = 2$) used in our experiments, an agent cannot see a hazard at Manhattan distance ≥ 3, so `min_dist=3` is information-theoretically infeasible. We therefore restrict our relational scenarios to $\texttt{min_dist} \in \{1, 2\}$. We further surface `min_dist` in the `info` dict and emit a warning when `min_dist` exceeds the partial-observation radius.

Other. We additionally migrated the codebase from `gym` to `gymnasium` (five-tuple `step`, `reset(seed, options)`, and the `np_random.integers` API) and enriched the `info` dict with `constraint`, `max_violations`, and a `full_map` rendering for downstream analysis. These do not affect the experimental setup and are detailed in the released code.

E ES-CoT AND DYNASOR HYPERPARAMETERS

Answer sources. For base models, ES-CoT and Dynasor obtain intermediate answers by appending an answer-forcing suffix to the reasoning prefix and sampling a short probe completion (Table 5). For Behavior Cue Reasoning models, the answer at each observation point is the most recent `[answer]` completed before that point; no probe is generated. Observation points are identical for both answer sources: ES-CoT observes after each completed reasoning line containing "wait", and Dynasor observes every *interval* tokens. Probe completions are sampled with temperature 0.6, top-p 0.95, and top-k 20 (Qwen3-8B) or 40 (GLM-Z1-9B), with a fixed seed per trajectory, method, and position. Dynasor with *full-response* confidence rejects a probe containing any of {wait, hold, but, okay, no, hmm}; *answer-only* confidence uses only the extracted answer.

Table 5: Probe settings for base models.

Domain	Method	Probe suffix	Max probe tokens
AIME2025	ES-CoT	`\nThe final answer is`	64
AIME2025	ES-CoT (boxed)	Dynasor suffix below	20
AIME2025	Dynasor	`... Oh, I suddenly got the answer to the whole problem, **Final Answer**\n\n\[ \boxed{`	20
Cookingworld	ES-CoT	`\nThe final action is <action>`	24
Cookingworld	Dynasor	`... Oh, I suddenly got the answer to the whole problem, **Final Action**\n\n<action>`	24

Search grids. ES-CoT is parameterized by the minimum run-length jump $d_{\min}$, the significance level α, and a minimum stopping position. Dynasor is parameterized by the observation interval, the stability window (number of consecutive identical confident answers), a minimum stopping position, and the confidence filter. Each configuration is evaluated by causal replay over saved trajectories (Table 6); a disabled configuration (no early stopping) is always included.

Table 6: Hyperparameter search grids. Min. pos. is the earliest token position at which a stop is allowed.

Setting	ES-CoT	Dynasor
AIME2025, base	$d_{\min} \in \{1, 2, 3, 5, 8, 10, 15, 20, 30, 50\}$; $\alpha \in \{0.01, 0.025, 0.05, 0.1, 0.2\}$; min. pos. $\in \{0, 1024, 2048, 4096, 8192, 12288\}$; probe $\in$ {original, boxed}	interval $\in \{128, 256, 512, 1024, 2048\}$; window $\in \{2, 3, 4, 5, 8, 10, 12, 16, 24, 32, 48, 64, 96, 128\}$; min. pos. $\in \{0, 1024, 2048, 4096, 8192, 12288\}$; confidence $\in$ {full-response, answer-only}
AIME2025, Behavior Cues	$d_{\min} \in \{0, 1, 2, 3, 5, 8, 10, 12, 15, 20, 30, 50, 80, 128\}$; $\alpha \in \{0.005, 0.01, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\}$; min. pos. $\in \{0, 2048, 4096, 8192, 12288, 16384, 20480, 24576, 28672\}$	interval $\in \{128, 256, 512, 1024, 2048\}$; window $\in \{1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 24, 32, 48, 64, 96, 128, 160, 192, 256\}$; min. pos. as for ES-CoT; confidence = answer-only
Cookingworld, base and Behavior Cues	$d_{\min} \in \{1, 2, 3, 5, 8, 10, 15, 20\}$; $\alpha \in \{0.01, 0.025, 0.05, 0.1, 0.2\}$; min. pos. $\in \{0, 64, 128, 256, 512, 1024\}$	interval $\in \{32, 64, 128, 256\}$; window $\in \{2, 3, 4, 5, 6, 8, 10\}$; min. pos. $\in \{0, 64, 128, 256, 512, 1024\}$; confidence $\in$ {full-response, answer-only} (answer-only for Behavior Cues)

Selected configurations. Table 7 lists the configuration behind each entry of Table 3.

Table 7: Configurations reported in Table 3. ES-CoT: ($d_{\min}$, α, min. position). Dynasor: (interval, window, min. position, confidence), with F = full-response and A = answer-only.

Domain	Model	Method	Base	Behavior Cues
AIME2025	Qwen3-8B	ES-CoT	(2, 0.1, 8192)	(0, 0.8, 16384)
		Dynasor	(256, 2, 0, F)	(2048, 10, 0, A)
	GLM-Z1-9B	ES-CoT	(5, 0.2, 0), boxed probe	(5, 0.005, 0)
		Dynasor	(256, 16, 0, A)	(128, 1, 16384, A)
Cookingworld	Qwen3-8B	ES-CoT	(1, 0.2, 0)	no stopping
		Dynasor	(128, 6, 1024, A)	(256, 5, 0, A)
	GLM-Z1-9B	ES-CoT	(1, 0.01, 1024)	(1, 0.05, 0)
		Dynasor	(32, 5, 1024, A)	(256, 6, 0, A)

F Compute

We predict all experiments run end to end would take roughly a month on a cluster of A100 with 80GB of memory.

The primary bottleneck comes from the extended reasoning length allowed for the complex math problems in AIME. This necessitates a larger memory footprint for SFT and RL training.

Data generation for initial reasoning traces and subsequent Behavior Cue Reasoning traces can be parallelized across all gpus for an 8B or 9B model relatively quickly. Completions are far faster relatively, as the model is required to only generate a handful of tokens.

G Reporting

For AIME and Cookingworld baseline experiments, baselines results are reported as average over 3 seeds for both base model and for the Behavior Cue Reasoning model. For Hazardworld, only one seed is used as the seeds are meant to represent an explicit scenario set verified by the authors.

H Extended Societal Impacts

This work is motivated by scalable oversight: making reasoning models easier to monitor and intervene on before final outputs are produced. In beneficial deployments, Behavior Cues could help reduce wasted inference compute, expose decision points for external oversight, and prevent unsafe commitments in constrained environments.

However, the same interface creates risks when reasoning is mediated through APIs or other systems that hide intermediate tokens from end users. Because Behavior Cues can be parsed or enforced without being surfaced in the final response, a platform or intermediary could use them to silently steer model reasoning without the user's awareness. This may be beneficial when used for disclosed safety interventions, but it also raises transparency and consent concerns if cue-based control is applied invisibly or for objectives misaligned with the user.

Finally, Behavior Cues should not be interpreted as deterministic guarantees of model behavior. LLM generation is inherently stochastic: our training procedure creates strong priors that anchor cue tokens to specific behaviors, but these associations are not guaranteed to hold on every sample or in every domain. This is reflected in the varying degrees of cue compliance observed across our experiments. We therefore view Behavior Cues as an interface that can improve monitorability and controllability, not as a replacement for careful validation, adversarial testing, and clear disclosure when reasoning-control mechanisms are deployed.