ADAPTIVE SOFT THINKING FOR CONTROLLING CANDIDATE INFLUENCE ACROSS REASONING STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) reasoning in large language models (LLMs) proceeds through discrete token generation, requiring a single token to be selected at each step even when multiple alternatives remain plausible. Latent reasoning addresses this limitation through continuous intermediate representations, and Soft Thinking provides a training-free approach by constructing Concept Tokens as probability-weighted mixtures of multiple token candidates. However, our preliminary analysis shows that the effect of candidate integration varies across reasoning states and is partially reflected in the current state, motivating state-dependent control of Soft Thinking. Accordingly, we propose ST-Scheduler, a lightweight Soft Thinking scheduler that controls candidate influence through state-dependent Concept Width selection while keeping the underlying LLM frozen. We optimize ST-Scheduler with reinforcement learning with verifiable rewards, using final reasoning outcomes to learn Concept Width decisions throughout reasoning. Across multiple reasoning models and mathematical and scientific reasoning benchmarks, ST-Scheduler outperforms standard CoT, fixed-width Soft Thinking, and existing training-free Soft Thinking methods, while achieving higher solution coverage with fewer sampled trajectories. Code is available at `https://anonymous.4open.science/r/ST-Scheduler-FC3F`.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong reasoning capabilities across complex tasks such as mathematics and logical problem solving through explicit Chain-of-Thought (CoT) reasoning (Wei et al., 2022). Recent reasoning models further strengthen these capabilities by allocating additional computation to intermediate reasoning before producing a final answer (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025). However, each intermediate CoT step commits to a single token choice, so a single reasoning trajectory follows only one of the alternatives available at that step. To exploit alternatives that are not followed within a single trajectory, existing approaches generate multiple reasoning trajectories for the same query and aggregate, select, or search across them (Wang et al., 2022; Yao et al., 2023; Wang & Zhou, 2024; Hassid et al., 2025).

In contrast, latent reasoning represents intermediate reasoning in continuous space, allowing alternatives to be incorporated within a single reasoning trajectory (Hao et al., 2024; Chen et al., 2025; Shen et al., 2025; Xu et al., 2025). Among these approaches, Soft Thinking provides a training-free method that replaces a discrete token with a Concept Token constructed from a probability-weighted combination of top-k candidate token embeddings (Zhang et al., 2026). The resulting Concept Token is then used as the next reasoning input, allowing information from multiple candidate tokens to be carried into the following reasoning step through a single continuous representation.

Despite this advantage, Concept Tokens introduce a structural distribution mismatch for pretrained LLMs, which are trained on discrete token embeddings rather than their probability-weighted mixtures (Zhang et al., 2026; Zheng et al., 2025). Under this constraint, prior work has identified two limitations of Soft Thinking: candidate integration at high-confidence states can introduce perturbations, while candidate information often has limited influence on subsequent computation as reasoning tends to follow the top-1 path (Fu & Luo, 2026; Shi et al., 2026; Wu et al., 2026; Rizvi-Martel et al., 2026). Accordingly, existing training-free methods have refined the inference-time application

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.

of Soft Thinking through separate controls over *when* candidate information is integrated and *how* it is combined within the Concept Token (Shi et al., 2026; Fu & Luo, 2026).

However, we take a more fundamental view of these limitations: **the effect of candidate integration varies across reasoning steps.** This view raises a new research question: can this variation be anticipated from the current reasoning state and leveraged to control Soft Thinking more effectively? To investigate this question, we conduct a preliminary analysis that quantifies how candidate influence is distributed across reasoning steps and visualizes how the next-step computation differs across levels of candidate influence. We also find that changing candidate integration at only a single step can alter the subsequent reasoning trajectory and the final answer. Importantly, states where such outcome-changing effects occur are partially distinguishable using the model representations at the corresponding reasoning step. Taken together, these findings suggest a unified objective for addressing the limitations of Soft Thinking: controlling candidate influence across reasoning states.

Motivated by these findings, we propose ST-Scheduler, a lightweight module that adaptively controls candidate integration at each reasoning step according to the current reasoning state while keeping the underlying LLM frozen. The key idea is to treat Concept Width, conventionally used as a fixed hyperparameter in Soft Thinking, as a state-dependent control variable, where $n = 1$ corresponds to the embedding of a single discrete token and larger widths incorporate information from multiple candidate tokens. Since each Concept Width decision affects subsequent reasoning states, we optimize ST-Scheduler with a group-relative policy objective using verifiable final outcomes, allowing width decisions to be learned from the relative outcomes of complete reasoning trajectories.

We evaluate ST-Scheduler across mathematical and scientific reasoning tasks using multiple reasoning LLMs. Our experiments compare ST-Scheduler with conventional CoT and existing training-free Soft Thinking approaches to examine whether adapting candidate integration across reasoning states improves the effectiveness of Soft Thinking. We further analyze the learned Concept Width behavior, candidate influence, and solution coverage to understand how state-dependent control changes the use of candidate information throughout the reasoning process.

Our main contributions are summarized as follows:

- We present a unified view of the observed limitations of training-free Soft Thinking by focusing on how the effect of candidate integration varies across reasoning steps. Our preliminary analysis supports this view by showing that local changes in candidate integration can propagate to final reasoning outcomes, while states exhibiting such outcome-changing effects are partially distinguishable using the model representations available at the corresponding step.
- We propose ST-Scheduler, a lightweight module that adapts Soft Thinking to the current reasoning state through state-dependent Concept Width selection while keeping the underlying LLM frozen. Rather than training the LLM to acquire Soft Thinking-specific reasoning patterns, ST-Scheduler learns how to adapt the use of Soft Thinking across reasoning states through a group-relative policy objective over verifiable reasoning outcomes.
- Through extensive experiments across multiple reasoning benchmarks, model families, and model scales, we demonstrate the effectiveness of adapting candidate integration across reasoning states. We further analyze the learned Concept Width behavior, candidate influence, and solution coverage to understand how state-dependent control changes the use of candidate information.

2 Related Work

2.1 Explicit Chain-of-Thought Reasoning

Chain-of-Thought (CoT) reasoning improves complex problem solving by generating explicit intermediate reasoning steps (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022). Subsequent work extends explicit reasoning through multi-trajectory sampling and search, alternative decoding, and reinforcement learning with verifiable rewards (RLVR) (Wang et al., 2022; Yao et al., 2023; Besta et al., 2024; Gandhi et al., 2024; Wang & Zhou, 2024; Shao et al., 2024; Guo et al., 2025). Despite these advances, explicit reasoning remains constrained to sequences of discrete token decisions, motivating approaches that intermediate computation beyond discrete language representations.

2.2 LATENT REASONING

Latent reasoning extends intermediate computation beyond discrete tokens using continuous representations (Chen et al., 2025; Deng et al., 2023; 2024; Yang et al., 2024; Shalev et al., 2024; Zhu et al., 2026). Training-based approaches learn continuous reasoning states through hidden-state recurrence, self-distillation, or latent compression, whereas training-free approaches modify inference without updating the pretrained LLM (Hao et al., 2024; Shen et al., 2025; Su et al., 2025; Zhang et al., 2025; Xu et al., 2025). Among training-free methods, Soft Thinking constructs Concept Tokens as probability-weighted combinations of candidate token embeddings (Zhang et al., 2026; Zhuang et al., 2026). Recent work identifies limitations such as perturbations from indiscriminate activation and limited propagation of candidate information, and addresses them by controlling when Soft Thinking is applied or how Concept Tokens are constructed (Wu et al., 2026; Rizvi-Martel et al., 2026; Fu & Luo, 2026; Shi et al., 2026). Other approaches instead train the underlying LLM to better operate with latent or soft-token representations (Butt et al., 2026; Tang et al., 2026; Zheng et al., 2025; Deng et al., 2026). In contrast, we keep the underlying LLM fixed and learn a separate Concept Width policy that adapts candidate integration across reasoning states.

3 METHODOLOGY

3.1 PRELIMINARY: EXPLICIT COT AND SOFT THINKING

Explicit CoT. Explicit CoT reasoning generates a sequence of discrete intermediate tokens before producing the final answer. Given a query q, let $r_{1:T} = (r_1, \dots, r_T) \in \mathcal{V}^T$ denote the intermediate reasoning trace over the vocabulary $\mathcal{V}$. At reasoning step t, the model predicts the next-token distribution $p_t(v) = p_\theta(v \mid q, r_{<t})$, $v \in \mathcal{V}$, and samples $r_t \sim p_t$. The reasoning phase terminates when an end-of-thinking token is generated, after which the final answer $a_{1:U}$ is decoded autoregressively. While explicit CoT provides an interpretable reasoning trace, each step commits to a single token from p_t, discarding information carried by the remaining candidates.

Soft Thinking. Soft Thinking avoids this discrete commitment by replacing the sampled reasoning token r_t with a continuous Concept Token $\mathbf{c}_t$. Given the token embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and token embedding $\mathbf{e}_v = \mathbf{E}[v]$, the Concept Token is constructed from the next-token distribution as

$$r_t \leftarrow \mathbf{c}_t = \sum_{v \in \mathcal{V}} p_t(v) \cdot \mathbf{e}_v. \tag{1}$$

The Concept Token construction is motivated by the following approximation, which replaces marginalization over all vocabulary paths with their probability-weighted embedding mixture:

$$\sum_{v \in \mathcal{V}} p_t(v)\, p_\theta(\cdot \mid q, r_{<t}, \mathbf{e}_v) \approx p_\theta\left(\cdot \mid q, r_{<t}, \sum_{v \in \mathcal{V}} p_t(v) \mathbf{e}_v\right) = p_\theta(\cdot \mid q, r_{<t}, \mathbf{c}_t)\,. \tag{2}$$

The resulting Concept Token is fed back to the model as the input for the next reasoning step. Once the end-of-thinking token becomes the most probable prediction, the reasoning phase terminates and the model switches to standard discrete decoding for the final-answer generation process.

Concept Width. In practice, Soft Thinking constructs each Concept Token from a truncated set of top-probability candidates rather than the entire vocabulary. We refer to the number of retained candidates as the *Concept Width* n and denote the corresponding top-n candidate set by $\mathcal{V}_t^{(n)}$. The width-specific Concept Token is constructed as

$$\mathbf{c}_t^{(n)} = \sum_{v \in \mathcal{V}_t^{(n)}} \hat{p}_t^{(n)}(v) \cdot \mathbf{e}_v, \qquad \hat{p}_t^{(n)}(v) = \frac{p_t(v)}{\sum_{u \in \mathcal{V}_t^{(n)}} p_t(u)}. \tag{3}$$

When $n = 1$, $\mathbf{c}_t^{(1)}$ reduces to a single discrete token embedding, whereas $n > 1$ integrates information from multiple candidate tokens into the Concept Token. Thus, Concept Width determines the degree of candidate integration at each reasoning step. Existing Soft Thinking typically applies a fixed Concept Width throughout the reasoning process. This fixed treatment motivates a more fundamental question: how does the effect of candidate integration vary across reasoning steps? We examine this question through preliminary analysis in the following section.

3.2 PRELIMINARY ANALYSIS

Analysis Setup. To investigate whether differences in the effect of candidate integration across reasoning steps can be anticipated from the current reasoning state and leveraged for state-dependent control, we conduct a preliminary analysis using Qwen3-8B on GSM8K. We analyze 100 greedy reasoning trajectories, consisting of 50 correctly solved and 50 incorrectly solved problems. The analysis examines (1) how candidate influence varies across reasoning steps and affects next-step computation, (2) whether a single-step change in candidate integration alters the final reasoning outcome, and (3) whether such outcome-changing states can be distinguished from information available at the corresponding step. Further details are provided in Appendix C.5.

(1) Candidate Influence Across Reasoning Steps. To quantify how candidate integration affects the next-step computation, we compare the prediction induced by a Concept Token with those induced by its individual candidate tokens. For each reasoning step t, let $v_t^{(i)}$ denote the i-th ranked candidate under p_t. We consider the three next-step inputs $\mathcal{B}_t = \{\mathbf{e}_{v_t^{(1)}}, \mathbf{e}_{v_t^{(2)}}, \mathbf{c}_t^{(2)}\}$, corresponding to the top-1 candidate, top-2 candidate, and their candidate-integrated Concept Token, respectively. For each $\mathbf{b} \in \mathcal{B}_t$, we feed $\mathbf{b}$ from the same reasoning prefix $(q, r_{<t})$ and let $T(\mathbf{b}) = \mathrm{Top10}(p_{t+1}(\cdot \mid q, r_{<t}, \mathbf{b}))$ denote the final-layer top-10 prediction set.

We define candidate influence as how much a Concept Token that mixes multiple candidates changes the next-step prediction from the top-1 discrete-token prediction. Accordingly, we measure the overlap between the Concept Token prediction and each individual candidate prediction as

$$O_{\text{top-}i} = \frac{|T(\mathbf{c}_t^{(2)}) \cap T(\mathbf{e}_{v_t^{(i)}})|}{10}, \qquad i \in \{1, 2\}. \tag{4}$$

We use $O_{\text{top-1}}$ as an inverse measure of candidate influence: a higher overlap indicates that candidate integration leaves the next-step prediction close to the single-token path, whereas a lower overlap indicates a stronger influence from the integrated candidate information.

As shown in Figure 1, the resulting candidate influence varies across reasoning steps. Approximately 74.4% of reasoning states fall in the high-overlap region ($O_{\text{top-1}} \geq 0.9$), including 42.9% with complete top-10 overlap. This indicates that candidate integration produces little change in the next-step prediction at most reasoning states, consistent with prior observations that Soft Thinking often remains close to the dominant top-1 path (Wu et al., 2026; Rizvi-Martel et al., 2026). In contrast, 18.4% of states fall in the medium-overlap region ($0.5 \leq O_{\text{top-1}} \leq 0.8$), while 7.3% fall in the low-overlap region ($O_{\text{top-1}} \leq 0.4$), where candidate integration produces substantially larger changes. This distribution directly shows that **the effect of candidate integration varies across reasoning states.**

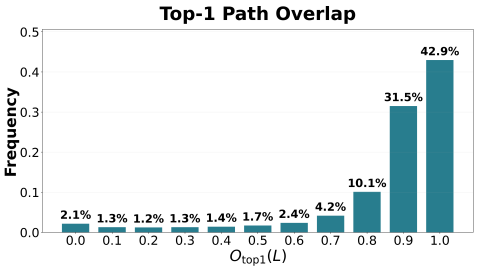


Figure 1: Distribution of final-layer top-1 path overlap $O_{\text{top-1}}$ across reasoning steps.

To examine how this state-wise variation in candidate influence emerges within next-step computation, we analyze intermediate representations across influence levels. We group states by final top-1 path overlap and apply Logit Lens (nostalgebraist, 2020) to trace across layers how the Concept Token path aligns with fixed top-1 and top-2 single-token paths, extending Eq. (4).

As shown in Figure 2, different levels of candidate influence correspond to distinct internal computation patterns. As candidate influence increases, the Concept Token path shifts from stronger alignment with the top-1 reference toward greater alignment with the top-2 reference across layers, indicating that information from the additional candidate is increasingly reflected in the next-step computation. **The effect of a Concept Token on next-step computation varies across reasoning states.** We next examine whether these local effects propagate through subsequent reasoning and alter the final answer.

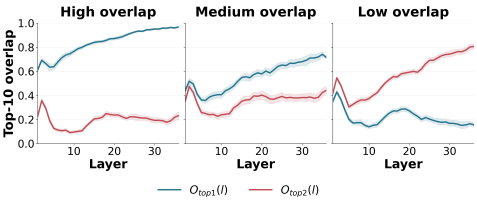


Figure 2: Layer-wise top-10 overlap between the Concept Token and single-token paths, aggregated over $N = 500$ states.

(2) Outcome Effects of Local Candidate Integration. To test whether the local effects observed in the next-step computation extend to the final reasoning outcome, we perform controlled single-step interventions on candidate integration. At each sampled reasoning state, we apply either $\mathbf{c}_t^{(2)}$ or $\mathbf{c}_t^{(3)}$ only at step t, yielding $p_{t+1}(\cdot \mid q, r_{<t}, \mathbf{c}_t^{(n)})$ for $n \in \{2, 3\}$, and continue discrete greedy decoding thereafter until the final response. We compare the final correctness of the resulting trajectories and further analyze representative outcome-changing cases in Appendix A.2.

We evaluate 1,000 intervention points in total, consisting of 500 states from originally incorrect greedy trajectories and 500 from originally correct trajectories. As shown in Table 1, although each intervention modifies only a single reasoning step, it changes final correctness in 6.4% of states from originally incorrect trajectories and 2.6% from originally correct trajectories. More importantly, the two Concept Width interventions lead to different final outcomes in 4.8% and 2.4% of the respective groups, indicating that the downstream effect depends not only on whether candidate information is integrated, but also on the degree of integration. **These results identify a sparse but consequential subset of reasoning states where a local change in candidate integration can alter the final reasoning outcome, motivating state-dependent control of its degree.**

Table 1: Changes in final correctness under single-step Concept Token interventions. "Change" is measured relative to the original greedy reasoning trajectory.

Original	No change	$\mathbf{c}^{(2)}$ only	$\mathbf{c}^{(3)}$ only	Both change	Any change
Incorrect	93.6%	2.8%	2.0%	1.6%	**6.4%**
Correct	97.4%	1.8%	0.6%	0.2%	**2.6%**

(3) Predictability of Outcome Effects from the Current Reasoning State. To assess whether the effect of candidate integration can be anticipated from the current reasoning state, we analyze pre-intervention model representations. We consider the top-N candidate logits $\mathbf{z}_t^{1:N}$, candidate embeddings $\mathbf{e}_t^{1:N}$, Concept Tokens $\mathbf{c}_t^{1:N}$, the final-layer hidden representation $\mathbf{h}_t^L$, and the layer-averaged hidden representation $\mathbf{h}_t^{1:L}$. For each signal, we fit an ℓ_2-regularized logistic regression probe to distinguish outcome-changing from interventions. Further details are provided in Appendix C.5.

Figure 3 reports AUROC for each probe, where higher values indicate better discrimination between outcome-changing and unchanged interventions, and 0.5 corresponds to random guessing. The candidate-level signals show limited and direction-dependent discrimination: the output logits yield AUROC below 0.5, while the candidate embeddings and Concept Tokens remain inconsistent across the two outcome-change directions. In contrast, the current hidden representations provide clearer predictive signals, with the layer-averaged representation achieving the highest AUROC among the tested representations. **These results suggest that the effect of candidate integration is partially reflected in the model's internal representation at the current reasoning step.** Motivated by this finding, we next develop a state-dependent Concept Width policy for controlling candidate integration.

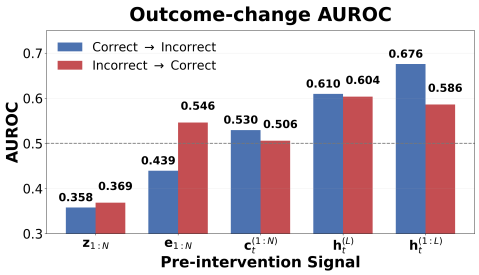


Figure 3: Comparison of AUROC across pre-intervention representations for distinguishing final correctness changes under single-step Concept Token interventions.

3.3 ST-Scheduler: State-Dependent Concept Width Selection

To control candidate influence across reasoning states, we introduce ST-Scheduler, a lightweight Soft Thinking scheduler that determines the Concept Width at each reasoning step while keeping the underlying LLM frozen. Motivated by our preliminary findings that the effect of a Concept Token varies across reasoning states and is partially predictable from the Concept Token and hidden representations at the current step, ST-Scheduler uses these signals to select the Concept Width. As illustrated in Figure 4, ST-Scheduler controls when to activate Soft Thinking and how to construct the Concept Token by learning a state-dependent policy over Concept Widths.

Let N denote the maximum Concept Width and $\mathcal{L}$ the set of LLM layers. For each layer l, ST-Scheduler projects the current hidden representation and the Concept Tokens constructed under different widths, concatenates them, and computes a score for each Concept Width:

$$\mathbf{s}_t^{(l)} = f_\phi\Big(\Big[\mathbf{W}_h\mathbf{h}_t^{(l)}; \mathbf{W}_c\mathbf{c}_t^{(1)}; \ldots; \mathbf{W}_c\mathbf{c}_t^{(N)}\Big]\Big), \qquad \mathbf{s}_t^{(l)} \in \mathbb{R}^N, \tag{5}$$

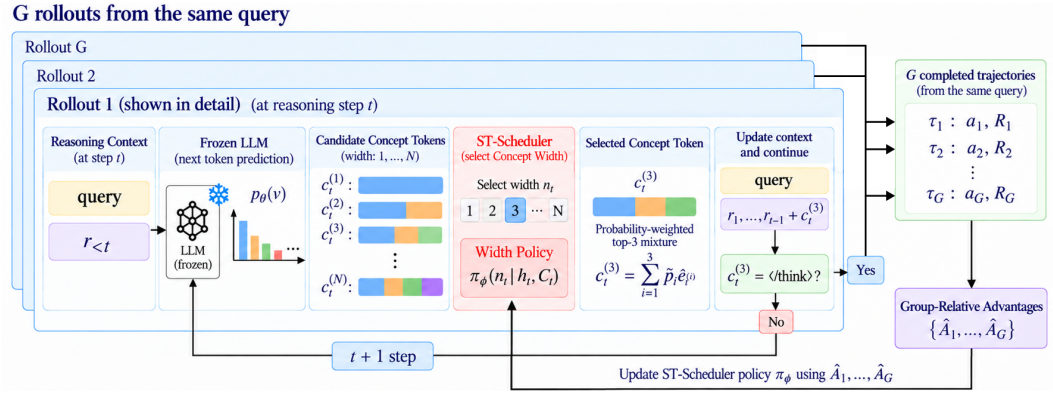


Figure 4: Overview of ST-Scheduler. At each reasoning step, the scheduler uses the current hidden representations and width-specific Concept Tokens to produce a policy over Concept Widths. The selected width determines the Concept Token used as the next reasoning input, while the underlying LLM remains frozen. The scheduler is optimized from group-relative verifiable outcomes.

where $\mathbf{W}_h$ and $\mathbf{W}_c$ are learnable projection matrices and f_ϕ is a lightweight MLP. Then, ST-Scheduler aggregates the layer-wise scores to predict a distribution over Concept Widths:

$$\bar{\mathbf{s}}_t = \frac{1}{|\mathcal{L}|}\sum_{l\in\mathcal{L}} \mathbf{s}_t^{(l)}, \qquad \pi_{\phi,t}(n) = \frac{\exp(\bar{s}_{t,n})}{\sum_{j=1}^{N}\exp(\bar{s}_{t,j})}, \quad n \in \{1,\dots,N\}. \tag{6}$$

The width distribution $\pi_{\phi,t}(n)$ serves as the state-dependent control signal for Soft Thinking. By adjusting this distribution across reasoning steps, ST-Scheduler controls when to activate Soft Thinking and how to construct the Concept Token according to the current reasoning state.

3.4 ST-Scheduler: Policy Optimization with Verifiable Rewards

To learn state-dependent Concept Width decisions from their effects on the final reasoning outcome, we optimize ST-Scheduler with reinforcement learning with verifiable rewards (RLVR). Rather than assigning an explicit target width at each reasoning step, we allow different width decisions to produce different reasoning trajectories and train the scheduler from their relative final outcomes. This formulation matches the sequential nature of Concept Width selection, where each decision changes the next reasoning input and consequently affects the states and decisions that follow.

For each query q, the rollout policy $\pi_{\phi_{\text{old}}}$ samples a Concept Width at each reasoning step, producing a group of G reasoning trajectories. Each trajectory receives a verifiable reward R_i based on its final answer, and its group-relative advantage is computed as $\hat{A}_i = (R_i - \mu_R)/\sigma_R$, where μ_R and σ_R denote the mean and standard deviation of rewards within the group. Since ST-Scheduler acts through the categorical Concept Width decision $n_{i,t}$, we optimize its width-selection policy using the standard clipped GRPO objective (Shao et al., 2024), with the policy ratio $\rho_{i,t} = \pi_{\phi,t}(n_{i,t})/\pi_{\phi_{\text{old}},t}(n_{i,t})$:

$$\mathcal{L}(\phi) = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\Big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t},1-\epsilon_c,1+\epsilon_c)\hat{A}_i\Big)\right]. \tag{7}$$

Through this objective, ST-Scheduler learns a state-dependent Concept Width policy from relative trajectory outcomes. Throughout optimization, only ST-Scheduler is updated, while the underlying LLM remains frozen. Thus, learning is restricted to how Soft Thinking is applied across reasoning states rather than modifying the reasoning model itself. Since the scheduler has no pretrained reference policy to preserve, we omit an additional reference-policy KL term.

4 Experiments

4.1 Experimental Settings

Models and Benchmarks. We evaluate ST-Scheduler across different model scales, model families, and reasoning domains. Experiments are conducted with Qwen3-1.7B and Qwen3-8B (Yang

Table 2: Pass@1 accuracy (%) across five reasoning benchmarks and three backbone models. Results are reported as the mean over five runs, with standard deviation in parentheses. The best result is highlighted in **bold**, and the second-best result is underlined.

Method	GSM8K	MATH500	GPQA	AIME24	AIME25	Avg.
			Qwen3-1.7B			
CoT (Wei et al., 2022)	90.31 (0.29)	91.08 (0.30)	38.38 (1.18)	43.33 (2.79)	34.67 (2.98)	59.55 (2.44)
Soft Thinking (Zhang et al., 2026)	90.16 (0.17)	90.16 (0.65)	35.55 (1.91)	38.00 (1.83)	35.33 (1.83)	57.84 (1.28)
SwiR (Shi et al., 2026)	90.58 (0.34)	91.48 (1.19)	39.29 (1.53)	46.67 (8.50)	36.00 (4.94)	60.80 (3.30)
SeLaR (Fu & Luo, 2026)	90.47 (0.52)	91.80 (1.07)	38.48 (1.15)	47.33 (4.35)	36.67 (5.27)	60.95 (2.47)
ST-Scheduler	**91.20 (0.27)**	**92.43 (0.98)**	**39.81 (0.75)**	**50.67(2.79)**	**38.67 (3.80)**	**62.55 (1.72)**
			Qwen3-8B			
CoT (Wei et al., 2022)	95.63 (0.35)	96.64 (0.38)	59.70 (1.15)	72.67 (4.35)	66.66 (4.71)	78.26 (2.19)
Soft Thinking (Zhang et al., 2026)	95.77 (0.37)	96.70 (0.48)	58.58 (1.43)	72.00 (2.98)	60.67 (3.65)	76.74 (1.78)
SwiR (Shi et al., 2026)	95.68 (0.45)	96.80 (1.09)	59.90 (1.91)	73.33 (4.08)	68.00 (4.47)	78.4 (2.40)
SeLaR (Fu & Luo, 2026)	95.40 (0.66)	96.75 (1.25)	59.98 (1.47)	74.00 (3.65)	68.66 (5.06)	78.96 (2.42)
ST-Scheduler	**95.79 (0.28)**	**97.18 (0.83)**	**60.81 (1.54)**	**75.33 (3.65)**	**69.33 (2.72)**	**79.69 (1.85)**
			DeepSeek-R1-Distill-Llama-8B			
CoT (Wei et al., 2022)	89.72 (0.56)	90.08 (0.93)	39.53 (5.21)	36.67 (7.07)	26.00 (7.23)	56.40 (4.20)
Soft Thinking (Zhang et al., 2026)	89.42 (0.64)	88.88 (4.21)	38.36 (4.21)	37.33 (4.94)	25.33 (5.05)	55.86 (3.81)
SwiR (Shi et al., 2026)	89.97 (0.66)	90.16 (3.57)	41.01 (3.15)	40.67 (8.63)	28.67 (5.58)	58.09 (4.32)
SeLaR (Fu & Luo, 2026)	89.82 (0.80)	90.36 (1.41)	40.33 (5.35)	41.33 (7.68)	28.00 (6.91)	57.97 (4.43)
ST-Scheduler	**90.31 (0.71)**	**91.16 (1.12)**	**41.75 (3.53)**	**43.33 (4.08)**	**30.67 (4.95)**	**59.44 (2.88)**

et al., 2025) and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). We conduct experiments on five reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024), AIME 2024 (Zhang & Math-AI, 2024), and AIME 2025 (Zhang & Math-AI, 2025) for mathematical reasoning, and GPQA-Diamond (Rein et al., 2023) for scientific reasoning.

Baselines. To compare state-dependent Concept Width selection with standard discrete reasoning and existing training-free Soft Thinking approaches, we evaluate CoT, Soft Thinking, SwiR, and SeLaR (Wei et al., 2022; Zhang et al., 2026; Shi et al., 2026; Fu & Luo, 2026). All Soft Thinking-based methods operate on the same frozen reasoning LLM: existing methods control candidate integration using predefined inference-time rules, whereas ST-Scheduler learns only an external state-dependent control policy. Thus, the comparison focuses on how Soft Thinking is controlled under a fixed underlying reasoning model rather than on modifying the reasoning capability of the backbone.

Metrics. We use Pass@1 as the primary metric for reasoning performance. All of the Pass@1 results are averaged over five independent runs with their standard deviation. For solution coverage, we additionally report Pass@k. For each query, given n trajectories with c correct solutions, Pass@k estimates the probability that at least one correct solution is found among k selected trajectories:

$$\text{Pass@}\,k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}. \tag{8}$$

Implementation Details. ST-Scheduler is trained on DAPO-Math-17K (Yu et al., 2026) using verifiable final-answer rewards with $G = 4$ rollout trajectories per query, and the checkpoint with the highest validation accuracy is selected for evaluation. For fair comparison, all baselines are reproduced using their official implementations and reported hyperparameters on NVIDIA A100 GPUs. All methods use the same decoding settings with temperature 0.6, top-$p = 0.95$, top-$k = 20$, and min-$p = 0.0$. Additional experimental setup details are provided in Appendix C.

4.2 Main Results

To evaluate whether controlling candidate influence across reasoning states improves reasoning performance, Table 2 reports Pass@1 across mathematical and scientific reasoning benchmarks. Across the evaluated models and benchmarks, ST-Scheduler achieves the strongest overall performance among the compared methods while also showing low variance across independent runs. This trend

holds across different model scales, model families, and reasoning domains, indicating that the benefit of state-dependent Concept Width control is not limited to a particular backbone or benchmark.

These results support our central claim that effectively leveraging Soft Thinking requires controlling candidate integration across reasoning states rather than applying a uniform rule throughout the reasoning process. By determining Concept Width from the current state, ST-Scheduler jointly adjusts whether additional candidates are incorporated and how broadly they are integrated. The reduced variance further indicates that the performance gains are more stable across independent runs. Since the underlying LLM remains fixed, the observed gains show that Soft Thinking can be better leveraged by adapting its use without modifying the reasoning model itself.

4.3 Analysis of ST-Scheduler's Concept Width Control Patterns

To understand how ST-Scheduler controls Soft Thinking after training, we analyze its Concept Width selection behavior through three questions: what Concept Width selection pattern emerge, when these selections are effective, and how broadly candidate information is integrated. All analyses are conducted on the same data used in Section 3.2, with further details provided in Appendix C.

What Concept Width Selection Patterns Does ST-Scheduler Learn? To examine how ST-Scheduler adjusts Concept Width across reasoning states with different uncertainty levels, we compute the next-token entropy $H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$ and group reasoning states into intervals of width 0.5 according to H_t. As shown in Figure 5, lower-entropy states are concentrated on smaller Concept Widths, with $n = 1$ dominating in more confident reasoning states. At higher entropy levels, the distribution shifts toward broader widths, indicating a greater tendency to integrate additional candidates. This tendency follows the same general direction as confidence-based activation strategies, which preserve discrete reasoning at high-confidence states and apply Soft Thinking under higher uncertainty (Shi et al., 2026; Fu & Luo, 2026). At the same time, each entropy interval retains selections across multiple widths rather than collapsing to a single choice. **Thus, ST-Scheduler learns a Concept Width selection pattern that follows the general tendency of confidence-based activation, while using Concept Width to jointly control when Soft Thinking is applied and how broadly candidate information is integrated.**

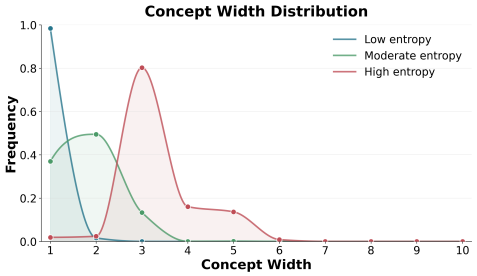


Figure 5: Distribution of selected Concept Widths across reasoning states grouped by next-token entropy in intervals of 0.5.

When Are Concept Width Selections Effective? To examine when Concept Width selections contribute to effective reasoning, we revisit the outcome-changing states identified in Section 3.2. We define the desired width decision according to the intervention outcome: $n = 1$ for correct-to-incorrect cases, where avoiding candidate integration preserves the correct answer, and $n > 1$ for incorrect-to-correct cases, where candidate integration leads to a correct answer. For each state, we compute the sampling probability assigned to the desired decision under $\pi_{\phi,t}$ and report its average.

As shown in Table 3, ST-Scheduler assigns an average sampling probability of 90.7% to $n = 1$ when candidate integration changes a correct outcome into an incorrect one, and 29.8% to $n > 1$ when broader integration changes an incorrect outcome into a correct one. **The learned policy therefore assigns high probability to $n = 1$ at states where candidate integration is harmful, while retaining non-negligible probability on broader widths at beneficial intervention states.** The asymmetric behavior is consistent with the long-tailed uncertainty distribution of reasoning trajectories, where low-entropy states constitute the majority and exploratory states occur less frequently (Fu & Luo, 2026). Further strengthening the latter behavior requires more informative local credit, but the sequential dependence between reasoning states makes dense step-level credit assignment challenging for adaptive Soft Thinking control.

Table 3: Average desired-width sampling probability at outcome-changing states.

Outcome Change	Desired Width	Selection Probability
Correct → Incorrect	$n = 1$	**90.7%**
Incorrect → Correct	$n > 1$	**29.8%**

How Broadly Does ST-Scheduler Integrate Candidate Information? To examine how ST-Scheduler determines the breadth of candidate integration, we measure the cumulative probability mass covered by the selected Concept Width. For sorted probabilities $p_{t,(1)} \geq p_{t,(2)} \geq \cdots$, let $M_t(k) = \sum_{j=1}^{k} p_{t,(j)}$ and compare $M_t(n_t - 1)$, $M_t(n_t)$, and $M_t(n_t + 1)$ around the selected width.

As shown in Figure 6, the selected widths typically cover approximately 90–97% of the candidate probability mass. Using one fewer candidate leaves a larger portion of the distribution uncovered, whereas adding one more contributes relatively little additional mass. Although no explicit probability-mass threshold is optimized for Concept Width selection, ST-Scheduler tends to retain most of the high-probability candidate mass while avoiding deeper low-probability tails. **This behavior is analogous to top-p filtering but serves a different purpose: top-p controls which candidates remain eligible for generation sampling, whereas ST-Scheduler learns to what extent those candidates should be integrated into the Concept Token at each reasoning state.**

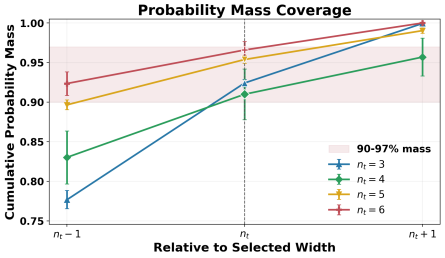


Figure 6: Cumulative probability mass around the selected Concept Width n_t.

Table 4: Pass@k results on AIME2024 and AIME2025 with Qwen3-1.7B varying $k \in \{1, 4, 8, 16\}$.

	AIME 2024				AIME 2025			
Method	Pass@1	Pass@4	Pass@8	Pass@16	Pass@1	Pass@4	Pass@8	Pass@16
CoT	43.33	64.14	71.97	**80.00**	34.67	47.41	54.77	**63.33**
CoT+GRPO	46.70	66.75	75.25	**80.00**	36.67	47.25	53.33	**63.33**
Soft Thinking	38.00	56.71	64.21	70.00	35.33	44.31	49.48	56.67
SwiR	46.67	67.85	74.78	**80.00**	36.00	47.83	54.82	**63.33**
SeLaR	47.33	63.51	69.76	76.67	36.67	47.36	54.87	60.00
ST-Scheduler	**50.67**	**69.88**	**76.45**	**80.00**	**38.67**	**50.14**	**55.98**	**63.33**

4.4 Solution Coverage

To examine whether state-dependent Concept Width control improves solution coverage beyond single-trajectory performance, Table 4 reports Pass@k results. Across both benchmarks, ST-Scheduler achieves the highest Pass@1, Pass@4, and Pass@8, while several methods reach the same performance at Pass@16, indicating higher solution coverage under smaller sampling budgets.

Fixed-width Soft Thinking shows consistently lower coverage, which may reflect the accumulated effect of applying candidate integration throughout reasoning, including states where additional candidate information is not beneficial. SeLaR improves this behavior through selective activation, but its lower Pass@k may also reflect the effect of explicitly reducing top-1 influence when constructing Concept Tokens. In contrast, the higher coverage of ST-Scheduler suggests that adaptive Concept Width control provides a more effective balance between preserving and integrating candidate information across reasoning. Notably, ST-Scheduler also exceeds CoT+GRPO at smaller k and reaches the same Pass@16 ceiling, despite training only the external Scheduler while keeping the underlying LLM frozen. **Overall, these results show that adaptively controlling candidate integration in Soft Thinking can achieve higher solution coverage with fewer sampled trajectories.**

5 Conclusion

We present ST-Scheduler, a lightweight scheduler for controlling candidate influence across reasoning states while keeping the underlying LLM frozen. Our analysis shows that the effect of candidate integration varies throughout reasoning and that this variation is partially reflected in the hidden representations available at the current step. Building on these findings, ST-Scheduler treats Concept Width as a state-dependent control variable and learns when and how candidate information is integrated from verifiable reasoning outcomes. Across different models and reasoning benchmarks, ST-Scheduler consistently improves over fixed-width Soft Thinking and existing training-free control methods, while also achieving stronger solution coverage with fewer sampled trajectories. Overall, our results show that adapting the use of Soft Thinking to the current reasoning state provides an effective way to leverage candidate information without modifying the underlying reasoning model.

AI USE STATEMENT

We used LLMs to support manuscript drafting and language refinement, and to assist with code implementation for the experiments. All AI-assisted text was reviewed by the authors, and all AI-generated code was manually inspected and validated before being used in the experimental pipeline. The authors remain fully responsible for the final manuscript and all associated code.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. In *International Conference on Learning Representations*, volume 2026, pp. 114650–114675, 2026.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jingcheng Deng, Zihao Wei, Liang Pang, Junhong Wu, Shicheng Xu, Zenghao Duan, and Huawei Shen. Latent-grpo: Group relative policy optimization for latent reasoning. *arXiv preprint arXiv:2604.27998*, 2026.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Renyu Fu and Guibo Luo. Selar: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7073–7087, 2026.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

nostalgebraist. Interpreting GPT: the logit lens. LessWrong, August 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 17-02-2025.

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.

Michael Rizvi-Martel, Guillaume Rabusseau, and Marius Mosbach. The illusion of superposition? a principled analysis of latent thinking in language models. *arXiv preprint arXiv:2604.06374*, 2026.

Yuval Shalev, Amir Feder, and Ariel Goldstein. Distributional reasoning in llms: Parallel reasoning processes in multi-hop reasoning. *arXiv preprint arXiv:2406.13858*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. Swireasoning: Switch-thinking in latent and explicit for pareto-superior reasoning llms. In *International Conference on Learning Representations*, volume 2026, pp. 137060–137093, 2026.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex thinking: Reasoning via token-wise branch-and-merge. *arXiv preprint arXiv:2601.08808*, 2026.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, et al. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. In *International Conference on Learning Representations*, volume 2026, pp. 4533–4548, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 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*, 2025.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13318–13339, 2025.

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.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Advances in Neural Information Processing Systems*, 38:168990–169012, 2026.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *Advances in Neural Information Processing Systems*, 38:79931–79963, 2026.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Mixture of inputs: Text generation beyond discrete token sampling. *Advances in Neural Information Processing Systems*, 38:26363–26387, 2026.

A Additional Analysis

We provide additional analyses of state-dependent Concept Width control. We first compare ST-Scheduler against fixed-width Soft Thinking to examine whether a single globally favorable Concept Width is sufficient. We then present representative reasoning trajectories illustrating why the effect of candidate integration depends on the current reasoning state. Finally, we analyze how Concept Width preferences emerge across transformer layers.

A.1 Effect of Fixed Concept Width

A central motivation for ST-Scheduler is that the appropriate degree of candidate integration varies across reasoning states. We therefore examine whether its improvement could instead be obtained by choosing a better fixed Concept Width. Table 5 compares fixed-width Soft Thinking with $n \in \{3, 5, 10\}$ against the state-dependent width policy learned by ST-Scheduler.

Table 5: Effect of Concept Width on mathematical reasoning. Fixed-width Soft Thinking uses the same Concept Width throughout reasoning, while ST-Scheduler selects the width at each reasoning step. Results are reported as the mean over five runs, with standard deviation in parentheses. The best result is highlighted in **bold**, and the second-best result is underlined.

Method	GSM8K	MATH500	GPQA	AIME24	AIME25	Avg.
Soft Thinking ($n = 3$)	89.67 (0.16)	90.08 (0.70)	37.67 (1.41)	41.33 (3.80)	30.67 (1.49)	57.88 (1.51)
Soft Thinking ($n = 5$)	90.18 (0.26)	90.24 (0.82)	36.26 (1.76)	34.00 (1.49)	34.67 (1.83)	57.07 (1.23)
Soft Thinking ($n = 10$)	90.16 (0.17)	90.16 (0.65)	35.55 (1.91)	38.00 (1.83)	35.33 (1.83)	57.84 (1.28)
ST-Scheduler	**91.20 (0.27)**	**92.43 (0.98)**	**39.81 (0.75)**	**50.67 (2.79)**	**38.67 (3.80)**	**62.55 (1.72)**

The preferred fixed width varies across benchmarks. Among the fixed-width variants, $n = 5$ performs best on GSM8K and MATH500, $n = 3$ performs best on GPQA and AIME 2024, and $n = 10$ performs best on AIME 2025. Thus, no single Concept Width consistently provides the strongest fixed-width performance.

More importantly, ST-Scheduler outperforms all fixed-width variants across every evaluated benchmark, improving average accuracy from 57.88% for the strongest fixed-width configuration to 62.55%. The difference is particularly pronounced on AIME 2024, where ST-Scheduler reaches 50.67% compared with 41.33% for the strongest fixed-width result. These results indicate that the benefit of ST-Scheduler cannot be reduced to selecting a better global Concept Width; adapting the degree of candidate integration to individual reasoning states provides a consistent advantage over globally fixed candidate integration.

A.2 Case Studies

To better understand how candidate integration affects subsequent reasoning, we examine two representative outcome-changing states from the preliminary analysis. Figure 7 shows the reasoning prefix before intervention, the candidate tokens available at the intervention step, and the resulting continuations under different Concept Widths. The two examples illustrate complementary roles of candidate information: additional candidates can provide information that helps escape an incorrect reasoning path, but they can also introduce noise that disrupts an otherwise correct continuation.

In GSM8K Problem 30, the model has already recognized that its current interpretation of the relation between the boots and the two pairs of heels is inconsistent, but continues reconsidering the same interpretation without resolving the error. At the intervention step, the top-1 and top-2 candidates are `there` and `the`, while the third candidate is `maybe`. The narrower $\mathbf{c}_t^{(2)}$, constructed only from the first two candidates, does not resolve the ambiguity, and the subsequent trajectory continues to predict \$94. In contrast, $\mathbf{c}_t^{(3)}$ additionally incorporates the information associated with `maybe`, after which the continuation revisits the interpretation of the problem and identifies the correct relation that the two pairs of heels, costing \$99, are \$5 cheaper than the boots. The trajectory consequently derives $99 = B - 5$ and reaches the correct answer \$104. This example demonstrates a benefit of candidate information integration: with an appropriate Concept Width, information from

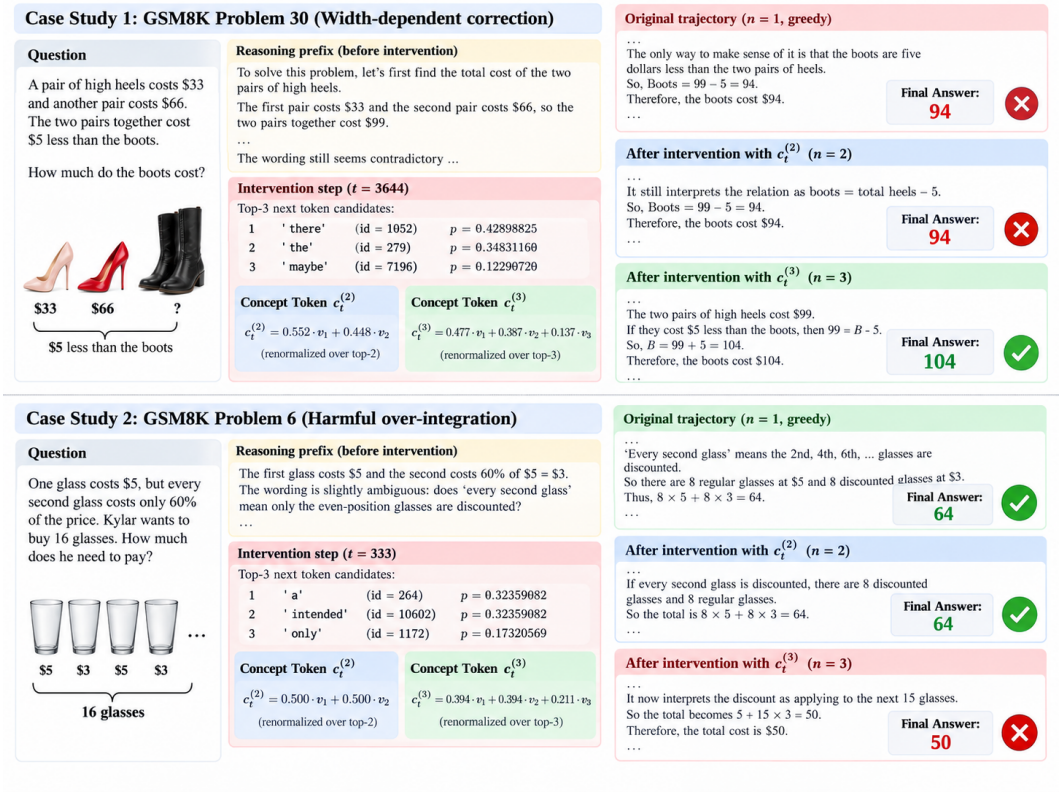


Figure 7: Two representative reasoning trajectories illustrating beneficial candidate integration and harmful over-integration under different Concept Widths.

a lower-ranked candidate can redirect a reasoning process that would otherwise remain trapped in an incorrect interpretation.

GSM8K Problem 6 exhibits the opposite effect. Before the intervention, the model is uncertain whether "every second glass" refers to alternating even-position glasses or to all glasses following the second. At this state, the top-1 and top-2 candidates are `a` and `intended`, while the third candidate is `only`. With $\mathbf{c}_t^{(2)}$, the subsequent reasoning resolves the ambiguity in favor of the correct interpretation: the 2nd, 4th, 6th, and other even-position glasses are discounted, yielding eight full-price and eight discounted glasses and the correct total of $64. However, the broader $\mathbf{c}_t^{(3)}$, which additionally incorporates `only`, shifts the continuation toward the incorrect interpretation that all 15 glasses after the first receive the discount. The resulting calculation becomes $5 + 15 \times 3 = 50$, turning an originally correct trajectory into an incorrect one. This case empirically demonstrates that additional candidate information can instead act as noise when it reinforces an inappropriate interpretation.

Together, these cases illustrate why candidate integration should be controlled according to the current reasoning state. A broader Concept Width can expose information that helps revise an incorrect trajectory, as in Problem 30, but the same broader integration can introduce distracting information and corrupt a correct trajectory, as in Problem 6. The effectiveness of Soft Thinking therefore depends not simply on preserving more candidates, but on selecting an appropriate degree of candidate integration for each reasoning state.

A.3 Layer-Wise Concept Width Preference

To examine how information relevant to Concept Width selection evolves across the LLM, we analyze the Scheduler prediction obtained from each transformer layer before layer-wise aggregation.

For each layer l, we apply the shared Scheduler head to obtain layer-specific width logits $\mathbf{z}_t^{(l)}$ and convert them into a width distribution:

$$\pi_{\phi,t}^{(l)}(n) = \mathrm{softmax}\left(\mathbf{z}_t^{(l)}\right)_n. \tag{9}$$

We then compute the expected Concept Width at each layer as

$$\mathbb{E}^{(l)}[n_t] = \sum_{n=1}^{N} n\, \pi_{\phi,t}^{(l)}(n). \tag{10}$$

Reasoning states are grouped according to the Concept Width selected by the final aggregated Scheduler policy, and the layer-specific expectations are averaged within each group.

As shown in Figure 8, states associated with different final Concept Width decisions exhibit increasingly distinct width preferences across layers. States assigned smaller widths maintain narrow preferences throughout the network, whereas states assigned broader widths progressively shift toward larger expected widths. The separation becomes particularly pronounced in later layers, indicating that information associated with the appropriate degree of candidate integration is progressively refined through the model's internal computation. At the same time, width-dependent differences are observable across multiple layers rather than emerging only at the final layer, supporting the use of layer-wise evidence aggregation in ST-Scheduler.

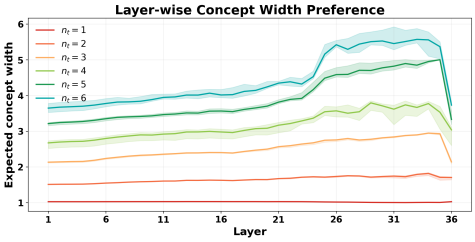


Figure 8: Layer-wise Concept Width preference. Each curve shows the expected Concept Width predicted from individual transformer layers for reasoning states grouped by the width selected by the ST-Scheduler.

B Additional Method Details

This section provides additional details of the notation and optimization procedure used by ST-Scheduler. While the main text describes the Scheduler architecture and its state-dependent Concept Width policy, we provide the complete training objective and rollout procedure here.

B.1 Notation

Table 6 summarizes the notation used throughout the paper. Vectors, including hidden representations and Concept Tokens, are denoted in bold.

B.2 GRPO Optimization

We train only the Scheduler parameters ϕ while keeping the underlying LLM parameters θ frozen. For each query q, the Scheduler samples a group of G reasoning trajectories. At reasoning step t of trajectory i, a Concept Width is sampled from the current policy,

$$n_{i,t} \sim \pi_{\phi,t}(\cdot), \tag{11}$$

and the corresponding Concept Token $\mathbf{c}_t^{(n_{i,t})}$ is used as the next reasoning input. The underlying LLM follows greedy decoding; therefore, stochasticity across trajectories is introduced through the sequence of Concept Width decisions rather than token sampling.

After each trajectory terminates, we evaluate its final answer using a task-specific verifier and assign a binary reward R_i. For a rollout group, the group-relative advantage is computed as

$$A_i = \frac{R_i - \mu_R}{\sigma_R + \epsilon}, \tag{12}$$

where μ_R and σ_R denote the mean and standard deviation of rewards within the group.

Table 6: Summary of notation used throughout the paper.

Symbol	Description
q	Input query or problem.
t	Reasoning step index.
l	Transformer layer index.
$\mathcal{L}$	Set of all transformer layers of the underlying LLM.
$\mathcal{V}$	Vocabulary of the underlying LLM.
$\mathbf{E} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d}$	Token embedding matrix of the frozen LLM.
$\mathbf{e}_v$	Embedding vector of token $v \in \mathcal{V}$.
θ	Parameters of the frozen underlying LLM.
$\mathbf{r}_{<t}$	Reasoning inputs preceding step t.
$p_t(v)$	Next-token probability $p_\theta(v \mid q, \mathbf{r}_{<t})$ at step t.
N	Maximum Concept Width.
n	Concept Width, where $n \in \{1, \ldots, N\}$.
$\mathcal{V}_t^{(n)}$	Set of the n highest-probability candidate tokens at step t.
$\mathbf{c}_t^{(n)}$	Concept Token constructed using Concept Width n.
$\mathbf{h}_t^{(l)}$	Hidden representation at reasoning step t from layer l.
$\mathbf{W}_h$	Learnable projection matrix for LLM hidden representations.
$\mathbf{W}_c$	Learnable projection matrix for Concept Tokens.
$\mathbf{x}_t^{(l)}$	Scheduler input constructed from $\mathbf{h}_t^{(l)}$ and all width-conditioned Concept Tokens.
f_ϕ	Lightweight MLP used for Concept Width prediction.
ϕ	Parameters of the Scheduler, including projection layers and the MLP.
$\mathbf{z}_t^{(l)} \in \mathbb{R}^N$	Concept Width logits predicted from layer l.
$\bar{\mathbf{z}}_t$	Mean Concept Width logits aggregated across transformer layers.
$\pi_{\phi,t}(n)$	Scheduler policy over Concept Widths at reasoning step t.
n_t	Concept Width sampled from $\pi_{\phi,t}$ at reasoning step t.
G	Number of reasoning trajectories sampled for each query during GRPO training.
R_i	Verifiable final-answer reward of trajectory i.
A_i	Group-relative advantage assigned to trajectory i.
$\rho_{i,t}$	Policy ratio for the width action taken at step t of trajectory i.
ϵ_c	PPO clipping coefficient.

For each width action taken during the rollout, we define the policy ratio

$$\rho_{i,t} = \frac{\pi_{\phi,t}(n_{i,t})}{\pi_{\phi_{\text{old}},t}(n_{i,t})}. \tag{13}$$

The Scheduler is optimized using the clipped group-relative objective

$$\mathcal{L}_{\text{GRPO}} = \mathbb{E}_{i,t}\left[\min\left(\rho_{i,t}A_i,\ \text{clip}\left(\rho_{i,t}, 1-\epsilon_c, 1+\epsilon_c\right) A_i\right)\right]. \tag{14}$$

Because the LLM remains frozen, optimization changes only how candidate information is integrated through Concept Width selection. The policy therefore learns from final reasoning outcomes without modifying the underlying token-generation distribution itself.

B.3 Effective Rollout Groups

To ensure that each policy update is supported by a non-zero group-relative learning signal, we adopt the Dynamic Sampling strategy of DAPO (Yu et al., 2026). For each query, we retain only rollout groups containing both correct and incorrect trajectories for policy optimization. When all G trajectories are correct or all are incorrect, the group has no reward variation and consequently yields zero group-relative advantages.

Formally, for a rollout group with rewards $\{R_i\}_{i=1}^G$, we define the group as effective when

$$\sigma_R > 0, \tag{15}$$

which, under binary rewards, is equivalent to

$$0 < \sum_{i=1}^{G} R_i < G. \tag{16}$$

Throughout training, more than 50% of the sampled rollout groups satisfy this condition and therefore provide effective optimization signals. This indicates that varying Concept Width alone, while keeping the underlying LLM decoding deterministic, produces sufficient outcome diversity for training the Scheduler.

Importantly, the reward is observed only after the full reasoning trajectory is completed. A single final outcome therefore provides supervision for all Concept Width decisions made along that trajectory. Since each width decision can alter subsequent reasoning states and influence later decisions, attributing the final outcome to a specific reasoning step remains challenging. This sparse trajectory-level credit assignment limits the precision with which beneficial local width decisions can be identified, helping explain the asymmetric behavior observed in Section 4.3.

B.4 Training Algorithm

Algorithm 1 summarizes the complete optimization procedure. For each query, we first sample G trajectories using the frozen LLM and the current Scheduler. Only groups containing reward variation are retained in the effective rollout buffer. The collected width actions are then reused for multiple clipped policy optimization iterations before new trajectories are generated.

Algorithm 1 Training ST-Scheduler with GRPO

Require: Frozen LLM p_θ, Scheduler π_ϕ, training set $\mathcal{D}$, maximum width N, group size G

for each training iteration **do**
  Sample a batch of queries from $\mathcal{D}$
  Set $\phi_{\text{old}} \leftarrow \phi$
  Initialize effective rollout buffer $\mathcal{B}_{\text{eff}} \leftarrow \emptyset$
  for each query q **do**
    for $i = 1, \ldots, G$ **do**
      for each reasoning step t **do**
        Compute p_t and hidden states $\{\mathbf{h}_t^{(l)}\}_{l \in \mathcal{L}}$ using the frozen LLM
        Construct $\{\mathbf{c}_t^{(n)}\}_{n=1}^{N}$ from p_t
        for $l \in \mathcal{L}$ **do**
          $\mathbf{x}_t^{(l)} \leftarrow [\mathbf{W}_h \mathbf{h}_t^{(l)}; \mathbf{W}_c \mathbf{c}_t^{(1)}; \ldots; \mathbf{W}_c \mathbf{c}_t^{(N)}]$
          $\mathbf{z}_t^{(l)} \leftarrow f_\phi(\mathbf{x}_t^{(l)})$
        end for
        $\bar{\mathbf{z}}_t \leftarrow |\mathcal{L}|^{-1} \sum_{l \in \mathcal{L}} \mathbf{z}_t^{(l)}$
        $\pi_{\phi,t} \leftarrow \mathrm{softmax}(\bar{\mathbf{z}}_t)$
        Sample $n_{i,t} \sim \pi_{\phi,t}$ and set $\mathbf{r}_t \leftarrow \mathbf{c}_t^{(n_{i,t})}$
      end for
      Evaluate final answer and obtain reward R_i
    end for
    Compute group reward statistics μ_R and σ_R
    if $\sigma_R > 0$ **then**
      Compute $A_i = (R_i - \mu_R)/(\sigma_R + \epsilon)$ for $i = 1, \ldots, G$
      Add the rollout group to $\mathcal{B}_{\text{eff}}$
    end if
  end for
  for $k = 1, \ldots, K_{\text{PPO}}$ **do**
    for each rollout group in $\mathcal{B}_{\text{eff}}$ **do**
      Compute $\rho_{i,t} = \pi_{\phi,t}(n_{i,t}) / \pi_{\phi_{\text{old}},t}(n_{i,t})$
      Update ϕ using $\mathcal{L}_{\text{GRPO}}$
    end for
  end for
end for
return π_ϕ

C Implementation Details

This section provides the training, baseline, dataset, and evaluation configurations used in our experiments.

C.1 Training Configuration

We train ST-Scheduler on DAPO-Math-17K using verifiable final-answer rewards (Yu et al., 2026). The last 64 problems of the training split are held out as a validation set and are excluded from Scheduler optimization. The checkpoint with the highest validation accuracy is used for evaluation.

The underlying reasoning LLM remains frozen throughout training, and all transformer layers of the backbone are used by the Scheduler. Both hidden representations and width-conditioned Concept Tokens are projected to 256 dimensions, and the Scheduler MLP uses a hidden dimension of 256. We set the maximum Concept Width to $N = 10$, giving a discrete action space of $\{1, \ldots, 10\}$.

For each prompt batch of 8 queries, we sample $G = 4$ trajectories per query and perform two PPO optimization iterations on each collected effective rollout group. We optimize the Scheduler using AdamW with a learning rate of 1×10^{-6}, PPO clipping coefficient $\epsilon_c = 0.2$, and the framework's default gradient clipping configuration.

During Scheduler training, the underlying LLM uses greedy decoding, while Concept Widths are sampled categorically from the Scheduler policy. The maximum response length during training is set to 20K tokens. At evaluation time, we increase the maximum response length to 32K tokens and cap the reasoning phase at 30K tokens. This provides sufficient space for long reasoning trajectories while reserving capacity for final-answer generation when reasoning becomes excessively long. Experiments are conducted on NVIDIA A100 GPUs.

Table 7 summarizes the training configuration.

Table 7: Training and implementation configuration of ST-Scheduler.

Hyperparameter	**Value**
Training dataset	DAPO-Math-17K
Validation set	Last 64 DAPO problems
Maximum Concept Width N	10
Width action space	$\{1, \ldots, 10\}$
Hidden projection dim.	256
Concept Token projection dim.	256
MLP hidden dim.	256
Prompt batch size	8
Group size G	4
PPO iterations	2
Learning rate	1×10^{-6}
PPO clipping ϵ_c	0.2
Optimizer	AdamW
Gradient clipping	1.0
Training LLM decoding	Greedy
Width sampling	Categorical
Training max response length	20K
Evaluation max response length	32K
Evaluation max thinking length	30K
Hardware	NVIDIA A100

C.2 Baseline Configuration

We compare ST-Scheduler against standard CoT and existing training-free Soft Thinking approaches. For the main evaluation, the baselines include CoT, fixed-width Soft Thinking, SwiR, and SeLaR (Wei et al., 2022; Zhang et al., 2026; Shi et al., 2026; Fu & Luo, 2026). The fixed-width

Soft Thinking baseline uses $n = 10$ throughout reasoning, while Appendix A.1 additionally evaluates $n \in \{3, 5, 10\}$. and SeLaR adapt the use of Soft Thinking during inference according to their respective control mechanisms. All Soft Thinking-based baselines keep the underlying LLM frozen and are reproduced using their official implementations and reported hyperparameters.

For the solution-coverage analysis, we additionally compare against a discrete-token CoT+GRPO baseline to distinguish state-dependent Soft Thinking control from directly optimizing the reasoning model. CoT+GRPO is trained with verifiable final-answer rewards using a group size of $G = 8$, while ST-Scheduler uses $G = 4$ trajectories and updates only the Scheduler.

C.3 Dataset Details

We train ST-Scheduler on DAPO-Math-17K and evaluate on five reasoning benchmarks spanning mathematical problem solving and knowledge-intensive scientific reasoning.

DAPO-Math-17K. DAPO-Math-17K provides mathematical reasoning problems with verifiable final answers and is used to optimize the Scheduler (Yu et al., 2026). We reserve the last 64 problems as a validation set and use the remaining examples for training. The validation examples are used only for checkpoint selection and are excluded from Scheduler optimization.

GSM8K. GSM8K evaluates multi-step mathematical reasoning in natural language (Cobbe et al., 2021). We use the official test set containing 1,319 grade-school math word problems that require explicit multi-step arithmetic reasoning. Dataset: `https://huggingface.co/datasets/openai/gsm8k`.

MATH500. MATH500 is a challenging subset of the MATH benchmark containing 500 competition-level mathematics problems (Lightman et al., 2024). It covers multiple mathematical domains, including algebra, geometry, number theory, and calculus, and requires non-trivial symbolic manipulation and multi-step deduction. Dataset: `https://huggingface.co/datasets/HuggingFaceH4/MATH-500`.

AIME 2024. AIME 2024 contains 30 problems from the 2024 American Invitational Mathematics Examination (Zhang & Math-AI, 2024). The problems require deep multi-step reasoning and precise numerical computation, with final answers represented as integers. Dataset: `https://huggingface.co/datasets/HuggingFaceH4/aime_2024`.

AIME 2025. AIME 2025 contains 30 newly released problems from the 2025 American Invitational Mathematics Examination (Zhang & Math-AI, 2025). It follows the same competition format as AIME 2024 while providing a more recent and challenging set for evaluating mathematical reasoning. Dataset: `https://huggingface.co/datasets/yentinglin/aime_2025`.

GPQA-Diamond. GPQA-Diamond is the most difficult split of GPQA and contains 198 expert-curated multiple-choice questions across scientific domains (Rein et al., 2023). The questions are designed to require advanced domain knowledge and rigorous multi-step reasoning rather than superficial pattern matching. Dataset: `https://huggingface.co/datasets/Idavidrein/gpqa`.

C.4 Evaluation Configuration

For fair comparison, all methods use matched prompts, generation budgets, and decoding configurations for each backbone. At evaluation time, we use temperature 0.6, top-$p = 0.95$, top-$k = 20$, and min-$p = 0.0$. The maximum response length is 32K tokens, with the reasoning phase capped at 30K tokens to ensure sufficient reasoning capacity while leaving room for final-answer generation.

Final answers are evaluated using task-specific verifiable correctness criteria. For GSM8K, MATH500, AIME 2024, and AIME 2025, correctness is determined from the extracted final numerical answer. For GPQA-Diamond, correctness is determined from the selected answer choice. Unless otherwise specified, Pass@1 results are averaged over five independent runs and reported with standard deviation.

C.5 Preliminary Analysis Configuration

The preliminary analysis in Section 3.2 is conducted with Qwen3-8B on the GSM8K test set. We construct a balanced set of 100 deterministic greedy reasoning trajectories, consisting of 50 correctly solved and 50 incorrectly solved problems. All reference trajectories are generated in bfloat16 with seed 42 and a maximum of 20,000 new tokens.

At each reasoning step, logits are scaled with temperature 0.6 and filtered using top-$k = 20$ and top-$p = 0.95$ before Concept Token construction. For the candidate-influence analysis, we compare next-step predictions produced from the top-1 token, top-2 token, and their integrated Concept Token from the same reasoning prefix.

For single-step counterfactual interventions, we reconstruct the exact reasoning prefix and KV cache at each selected state and replace only the current reasoning input with the corresponding Concept Token. Each resulting branch is then greedily decoded independently until EOS or the generation limit. The preceding reference trajectory is never modified.

For the Logit Lens analysis, we apply the model's final normalization and LM head to the hidden representation at each transformer layer and retain the top-10 projected tokens. Layer-wise overlap is measured against independently forwarded candidate-token paths. Curves aggregate 500 states per overlap group, and shaded regions correspond to problem-level bootstrap 95% confidence intervals computed from 5,000 resamples.

For the outcome-change analysis, we fit ℓ_2-regularized logistic regression probes using leave-one-problem-out cross-validation. Feature normalization is fitted only on training problems in each fold. The regularization coefficient is selected from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ using an inner held-out training problem, and AUROC is computed from pooled out-of-fold predictions. Correct-to-incorrect and incorrect-to-correct transitions are modeled separately.