ParentingLLM: Pacing Reasoning Maturity for Bidirectional Test-Time Scaling under Token Budgets

Anonymous ACL submission

Abstract

Reasoning LLMs under fixed token budgets often commit before verification finishes, or spend the budget on deliberation that never reaches a final answer. Existing inference-time controllers regulate reasoning *length*, which does not indicate whether the current prefix is ready to be finalized. We introduce PARENTINGLLM, an inference-time controller that paces *reasoning maturity*, defined as the prefix's readiness to commit given the remaining budget, against a user-specified target schedule, without fine-tuning the underlying LLM. The maturity estimate combines consumed budget with a forked *forced-commit probe*; the controller injects WAIT when the estimate runs ahead of the schedule and HURRY when it lags. On MATH-500, GPQA-Diamond, AIME 24+25, and GSM-Hard with three Gemma-4 models, PARENTINGLLM improves over prior inference-time baselines, with the largest gains on AIME 24+25 (-10.6 burst rate, $+5.0$ accuracy).

1 Introduction

Reasoning models such as OpenAI's o1/o3 series (OpenAI, 2024, 2025), DeepSeek-R1 (Guo et al., 2025), Qwen3 (Yang et al., 2025a), and Gemma 4 (Google, 2026) improve mathematical and scientific reasoning by generating longer chains of intermediate steps (Muennighoff et al., 2025). Allocating more compute at inference, i.e., *test-time scaling*, has become an alternative to scaling model size.

Longer reasoning, however, is not free. Each generated token extends autoregressive decoding, consumes additional GPU compute and KV-cache memory, and contributes directly to serving latency. Modern LLM serving systems therefore optimize scheduling and memory management under throughput and latency/SLO constraints (Yu et al., 2022; Kwon et al., 2023; Zhong et al., 2024), making uncontrolled generation length a practical

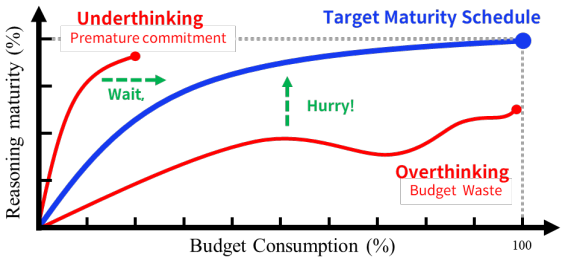


Figure 1: Two symmetric failures (red lines) of fixed-budget reasoning. Above: *underthinking*: premature commitment. Below: *overthinking*: budget wasted before finalization. Our method injects WAIT/HURRY cues to pace maturity along the target schedule (blue).

systems concern. A response that exceeds its budget may be truncated mid-reasoning and provide little utility. The deployment-relevant question is therefore not *how much* a model can think in the abstract, but *how effectively* it uses a fixed reasoning budget.

A reasoning LLM's decision to terminate involves two quantities: *what* the candidate answer is, and *whether the model is ready to commit to it*. Existing termination signals conflate these. Next-token entropy (Wang et al., 2026) peaks mid-trace, since a model performing late-stage verification can hedge cautiously even while its candidate is stable. Answer confidence from a forced trial answer (Yang et al., 2025b) captures certainty in the candidate, but reasoning models routinely hold a high-confidence candidate while continuing to self-verify. Inter-boundary consistency (Fu et al., 2024) flags when the candidate stops changing, but a stable candidate under ongoing verification reflects cautious decoding rather than readiness to stop. None of these asks whether the model itself would terminate cleanly if forced to. A model that is ready to commit will assign high probability to ending its turn once a candidate answer is in place; a model still mid-verification will not, even when the same candidate is on the table. This distinction is observ-

able from the model's own distribution and does not require a separate confidence estimate. We define a signal that asks exactly this: P_{EOT}, the end-of-turn probability after a forced-commit suffix.

This gap motivates the two failure modes we address (Fig. 1). *Underthinking* occurs when the model becomes willing to finalize too early, committing after only superficial verification and leaving useful budget unused. *Overthinking*, in our usage, is a trace-level budget failure: the model spends tokens on low-yield deliberation, redundant checks, or unproductive branches, so its commitment readiness fails to keep pace with the budget and the response risks truncation before finalization.[1] We argue that both failures are best understood as failures of *commitment readiness under budget*, not merely of length. We call this latent state *reasoning maturity*.

We propose PARENTINGLLM, an inference-time controller that paces reasoning maturity under a fixed token budget against a user-specified target schedule. At each reasoning boundary, PARENTINGLLM applies a *forced-commit probe* to read the end-of-turn probability after an elicited candidate answer, combines this signal with the current budget consumption to estimate reasoning maturity, and compares the estimate to the schedule. If the estimate exceeds the schedule, a WAIT cue is injected to delay commitment; if it lags, a HURRY cue is injected to apply finalization pressure. *Bidirectional test-time scaling* thus emerges from the sign of the maturity gap, without fine-tuning the underlying LLM.

Our contributions are as follows:

- We reformulate fixed-budget reasoning control as pacing *reasoning maturity*, and frame underthinking and overthinking as symmetric failures of maturity pacing.

- We propose PARENTINGLLM, a maturity-gap controller that realizes bidirectional test-time scaling via a forced-commit probe, implemented as a fork-and-discard path inside vLLM. The target maturity schedule exposes an operator-facing knob that shifts the controller between earlier commitment and longer verification, without fine-tuning.

[1] Prior work uses *overthinking* for post-solution surplus deliberation or within-step token redundancy; we use it in a trace-level, budget-relative sense.

2 Related Work

Budget-aware reasoning and inference-time steering. A growing line of work studies reasoning under explicit computational constraints (Alomrani et al., 2025). TALE-EP (Han et al., 2025) exposes the budget to the model and relies on its in-context self-regulation; Han et al. (2025) show that this works well when the LLM can produce accurate budget estimates (e.g., GPT-4o-mini), but the stated limit may still be treated as a soft preference and exceeded. Training-based methods such as BudgetThinker (Wen et al., 2025), L1 (Aggarwal and Welleck, 2025), and ThinkPrune (Hou et al., 2025) achieve tighter length control by updating model parameters, at the cost of portability across base models. Inference-time alternatives avoid retraining: s1-style budget forcing (Muennighoff et al., 2025) inserts WAIT tokens to extend reasoning, while Budget Guidance (Li et al., 2025) uses an auxiliary length predictor to align reasoning length with a budget. Related steering methods inject cues at reasoning transitions, including SmartSwitch's deepening prompts (Zhang et al., 2025), certainty-guided reflection suppression (Huang et al., 2026), and difficulty-adaptive slow-thinking (Shen et al., 2025). These methods primarily control reasoning length along a single direction. PARENTINGLLM is complementary along a different axis: it treats commitment readiness as the control target and exposes the desired pacing as an operator-facing schedule rather than a fixed length policy. The controller chooses WAIT, HURRY, or no intervention from the sign of a maturity gap, addressing cases where length control or self-regulation alone does not produce timely finalization.

Overthinking and underthinking. Sui et al. (2025) survey overthinking, where reasoning models generate verbose or redundant chains without improving answer quality, and Wang et al. (2025) study underthinking, where models prematurely abandon promising paths; SmartSwitch (Zhang et al., 2025) addresses the latter by injecting deepening prompts after backtracking. We build on this distinction in a budgeted form: underthinking as becoming willing to finalize before adequate verification, and overthinking as deliberation that fails to keep commitment readiness on pace with the budget. PARENTINGLLM unifies the two under maturity pacing, using a single controller to delay premature commitment or apply finalization pressure depending on the sign of the maturity gap.

Probe-based signals and uncertainty. Closer to our setting, training-free methods probe the model during generation to inform termination. DEER (Yang et al., 2025b) exits when an elicited trial answer is high-confidence; EAT (Wang et al., 2026) exits when next-token entropy stabilizes; Dynasor-CoT (Fu et al., 2024) monitors trial-answer consistency across transitions. A broader literature on LLM uncertainty, semantic entropy (Kuhn et al., 2023; Farquhar et al., 2024), relevance-weighted aggregation (Duan et al., 2024), semantic entropy probes (Kossen et al., 2024), and momentum-based uncertainty (Yan et al., 2025), estimates uncertainty over answers or reasoning steps, typically for post-hoc analysis or compute allocation. PARENTINGLLM extends this line in two ways: our forced-commit probe captures *commitment willingness*, whether the model would terminate cleanly if forced to answer now, rather than answer-level uncertainty; and the signal is used bidirectionally, prolonging reasoning when premature commitment is detected as well as stopping it when ready. Trial-answer changes between boundaries serve as an auxiliary prolongation signal (Sec. 4.4), related to Dynasor-CoT's consistency monitoring but used in the opposite direction.

3 Problem Definition

A reasoning LLM $\mathcal{M}$ generates a response y to a prompt x under a hard token budget B: generation is terminated once $|y| = B$, regardless of whether $\mathcal{M}$ has produced a final answer. Under this constraint, the two complementary failures of Fig. 1, *underthinking* and *overthinking*, are not reliably addressed by $\mathcal{M}$ itself: even when B is stated in the prompt, the model does not enforce it as a hard external constraint.

We seek a controller π that intervenes during inference to maximize expected answer quality under the budget, without training the parameters of $\mathcal{M}$:

$$\max_{\pi} \; \mathbb{E}_{x\sim\mathcal{D}}\big[Q(y \mid x, \pi, \mathcal{M})\big] \quad \text{s.t.} \quad |y| \le B, \tag{1}$$

where Q scores final answer quality (e.g., task accuracy) and π injects steering tokens (e.g., *Hurry up*, *Wait - let me think again*) at chosen reasoning boundaries. The controller may rely on lightweight auxiliary components fit offline, but $\mathcal{M}$ itself is never updated, and all generated tokens, including intervention cues, count toward B.

Because Q depends on the final answer, it is unobservable at inference time and cannot be optimized directly by π. We instead introduce an intermediate control variable, *reasoning maturity*: the degree to which the current prefix is ready to support a reliable final answer given the remaining budget. Maturity is a control-time notion of commitment readiness, not a quality measurement: a mature prefix can still produce a wrong answer. The controller's task is to keep maturity on pace with budget consumption at every boundary; the next section instantiates this via a forced-commit probe paced against a user-specified target schedule.

4 Proposed Method

4.1 Preliminaries

PARENTINGLLM controls generation at *reasoning boundaries* rather than at every token. A reasoning boundary is a natural segmentation point in the trace; in our experiments we use newline-delimited reasoning steps (Appendix A.2). At each boundary t_i, the controller observes the current prefix $y_{1:t_i}$ and the fractional budget consumption

$$u_i \;=\; t_i/B \;\in\; [0,1], \tag{2}$$

where t_i counts all output tokens generated so far.

4.2 Forced-Commit Probe

To estimate whether the current prefix is becoming ready to finalize, we run a side probe on a transient path: we append a commit suffix (e.g., "`The final answer is`") to $x \oplus y_{1:t_i}$. Because the task prompt instructs the model to wrap its final answer in `\boxed{...}`, this elicits a boxed candidate answer $\hat{a}_i$. Let $c_i = x \oplus y_{1:t_i} \oplus \texttt{commit suffix} \oplus \hat{a}_i$ denote the resulting probe context. We read the probability of the end-of-turn token immediately after the closing brace:

$$P_{\text{EOT},i} \;=\; \Pr\nolimits_{\mathcal{M}}\big(\text{EOT} \mid c_i\big). \tag{3}$$

Intuitively, $P_{\text{EOT},i}$ asks whether the model would terminate cleanly if forced to answer now. This is distinct from answer confidence: a model can be confident in a candidate answer while still continuing to verify or revise it. We therefore treat $P_{\text{EOT},i}$ as a probe-based measurement of *commitment willingness*.

The probe is a side computation: the boxed candidate answer $\hat{a}_i$ is used to compute $P_{\text{EOT},i}$ and to detect inter-boundary answer changes; it is never appended to the returned trace. Sec. 4.5 describes

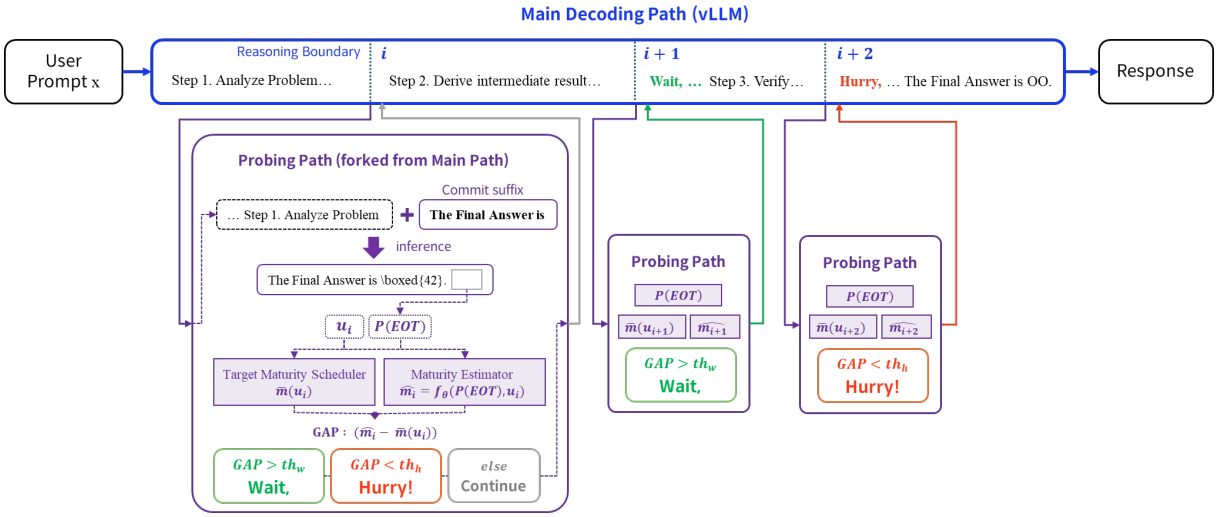


Figure 2: Inference-time pipeline of PARENTINGLLM. At each reasoning boundary, the runtime forks a transient probe path, appends a commit suffix, and reads $P_{\text{EOT},i}$; the maturity estimator f_θ maps $(u_i, P_{\text{EOT},i})$ to $\hat{m}_i$, and the controller selects WAIT, HURRY, or CONTINUE from the sign of the maturity gap $\hat{m}_i - \bar{m}(u_i)$. The probe path is discarded after the read; only the selected cue (if any) is written to the main path.

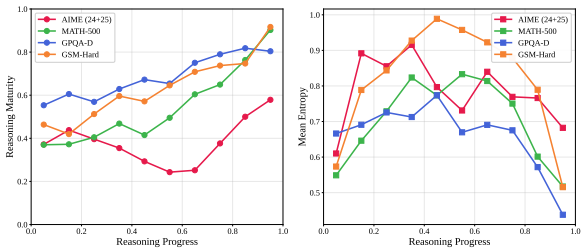


Figure 3: Per-benchmark trajectories as a function of reasoning progress (u, binned into deciles), averaged across traces. **Left:** estimated maturity $\hat{m}$ from P_{EOT}. **Right:** mean answer-token entropy.

how this is implemented as a fork-and-discard path in vLLM.

Figure 3 contrasts P_{EOT}-based maturity with next-token entropy, a common uncertainty surrogate in prior early-exit methods (Wang et al., 2026; Yan et al., 2025). Maturity rises monotonically with progress across all four benchmarks, while entropy is non-monotonic and peaks mid-trace. A signal that cannot distinguish a near-final prefix from a mid-deliberation one is unsuitable as a maturity proxy; we therefore use P_{EOT}, together with budget consumption, as the input to the maturity estimator.

4.3 Maturity Estimation and Target Schedule

The same forced-commit signal can mean different things early and late in the budget, so we combine it with u_i in a lightweight maturity estimator:

$$\hat{m}_i \;=\; f_\theta(u_i, P_{\text{EOT},i}) \;\in\; [0,1]. \qquad (4)$$

We instantiate f_θ as a gradient-boosted regressor (LightGBM (Ke et al., 2017)), fit once offline on boundary-level traces. Training labels are *judge-derived maturity scores*: an external LLM judge rates how ready a prefix is for reliable finalization under the remaining budget (prompt and rubric in Appendix B.2). These labels are used only to fit f_θ; at inference, the controller observes only u_i and $P_{\text{EOT},i}$. f_θ is fit *once per base model*, and the parameters of $\mathcal{M}$ are never updated.

We compare $\hat{m}_i$ against a user-specified *target maturity schedule* $\bar{m} : [0,1] \to [0,1]$, where $\bar{m}(u)$ specifies the desired commitment readiness at budget fraction u. The schedule is a pacing preference, not a measurement of problem-solving progress: a more concave $\bar{m}$ tolerates late maturity and biases the controller toward WAIT, while a flatter $\bar{m}$ demands earlier maturity and biases toward HURRY. Unless otherwise stated, we use the default

$$\bar{m}(u) = 1 - (1-u)^2, \qquad (5)$$

a neutral concave shape that matches the empirically observed maturity trajectory of unconstrained Gemma-4 traces (Fig. 3, left). Operators can shift along this axis without retraining f_θ or $\mathcal{M}$: $\bar{m}$ is a fixed function with no parameters fit from data.

4.4 Maturity-Gap Intervention

At each boundary, PARENTINGLLM computes the maturity gap

$$\Delta_i \;=\; \hat{m}_i - \bar{m}(u_i), \qquad (6)$$

Algorithm 1 Maturity-Gap Parenting Controller

Input: prompt x, model $\mathcal{M}$, estimator f_θ, schedule $\bar{m}$, budget B, threshold (τ_W, τ_H)
$y \leftarrow \emptyset$;
while $|y| < B$ **and** $\mathcal{M}$ has not terminated **do**
  Extend y by generating from $\mathcal{M}(x \oplus y)$ until the next reasoning boundary t_i
  $u_i \leftarrow t_i / B$
  $(P_{\text{EOT},i},\ \hat{a}_i) \leftarrow \text{PROBE}(\mathcal{M},\ x,\ y_{1:t_i})$
  $\hat{m}_i \leftarrow f_\theta(u_i, P_{\text{EOT},i})$
  $\Delta_i \leftarrow \hat{m}_i - \bar{m}(u_i)$
  if $\Delta_i > \tau_W$ **then**
    Append WAIT cue to y;
  else if $\Delta_i < -\tau_H$ **then**
    Append HURRY cue to y;
  else
    if $\hat{a}_i \neq \hat{a}_{i-1}$ **then**
      Append WAIT cue to y;
    else
      CONTINUE;
    end if
  end if
end while
return y

and selects an action among {CONTINUE, WAIT, HURRY} from its sign. WAIT appends a short cue that encourages further verification; HURRY appends a cue that encourages concise commitment; CONTINUE is a no-op (cue strings in Appendix A.3).

A positive gap signals a risk of *underthinking*: the model appears more commit-ready than desired at the current budget position, and may finalize before adequate verification. When $\Delta_i > \tau_W$, the controller injects WAIT. A negative gap signals a risk of *overthinking* in our budget-level sense: deliberation is failing to translate into sufficient commitment readiness relative to the consumed budget. When $\Delta_i < -\tau_H$, the controller injects HURRY. Otherwise it takes CONTINUE.

The thresholds $\tau_W, \tau_H \geq 0$ encode an asymmetric cost: a spurious WAIT mainly consumes extra tokens, while a spurious HURRY can force commitment on a still-unresolved problem. We therefore set $\tau_H > \tau_W$, using $\tau_W = 0.1$ and $\tau_H = 0.15$ in all experiments. We additionally cap the total number of cue injections per request at three. Beyond this, repeated cues begin to overwrite the model's in-progress reasoning rather than redirect it, degrading answer quality in preliminary runs.

As an auxiliary safeguard, the controller also caches the probe's candidate answer $\hat{a}_i$ (Sec. 4.2) across boundaries: when $\hat{a}_i \neq \hat{a}_{i-1}$ and the maturity-gap rule would otherwise select CONTINUE, the controller substitutes a WAIT cue. A change in the trial answer between consecutive boundaries indicates that the prefix is still being revised, and prolonging reasoning prevents finalization on an unstable hypothesis. The safeguard only fills in boundaries where the maturity gap is inconclusive and never overrides WAIT or HURRY from the main rule.

4.5 Runtime Integration

PARENTINGLLM is implemented as an inference-time controller inside a modified vLLM (Kwon et al., 2023) runtime (Fig. 2). For each user request, the runtime maintains a single main decoding path whose tokens form the returned response. At each reasoning boundary, the runtime forks the main path into a transient probe path, appends the commit suffix only on the probe path, and generates a boxed candidate answer to read $P_{\text{EOT},i}$. The probe path is then immediately discarded. Thus, the forced-commit answer is never shown to the user, never enters the returned trace, and never modifies the main decoding state.

Only the controller's selected action affects the returned response. If WAIT or HURRY is triggered, the corresponding cue is written to the main path and counts toward the response budget B. If CONTINUE is selected, the main path simply resumes decoding from its original pre-probe state. Probe tokens do not consume B because they are not part of the returned response, but their KV-cache footprint and forward-pass cost are real overheads, which we report in Sec. 5. The fork-and-discard design is compatible with vLLM-style continuous batching: transient probe paths can be scheduled alongside ordinary decoding paths and discarded after the probe is read.

5 Experiments

5.1 Experimental Setup

Models. We evaluate PARENTINGLLM on three reasoning-tuned LLMs from the Gemma-4 family at different scales: Gemma-4-31B-it, Gemma-4-26B-A4B-it, and Gemma-4-E4B-it (Google, 2026). This sweep tests whether the controller generalizes across model sizes within a consistent family.

Datasets. We use four reasoning benchmarks spanning mathematics, science, and clinical reasoning: **MATH-500** (Hendrycks et al., 2021), a 500-problem subset of MATH; **GPQA-Diamond** (Rein et al., 2023), graduate-level multiple-choice questions in physics, chemistry, and biology; **AIME 24+25** (60 problems combined), competition-level mathematics, and **GSM-Hard** (Gao et al., 2023), an adversarial variant of GSM8K with larger numeric operands. The first three benchmarks are split into calibration and evaluation portions and used to fit the maturity estimator f_θ and tune the thresholds (τ_W, τ_H); GSM-Hard is held out from calibration and used solely for evaluation, serving as an in-table generalization test. All tasks are evaluated with exact-match accuracy on the boxed final answer.

Token budgets. We evaluate at two budgets: $B \in \{1024, 2048\}$ output tokens. The lower budget stresses the controller's ability to force timely finalization; the higher budget tests whether the controller avoids spurious interventions when budget is less constraining.

Baselines. We compare against three inference-time baselines that, like PARENTINGLLM, do not modify the parameters of the base reasoning model: (i) **Vanilla**: the model decodes freely; if $|y| = B$ before termination, the response is hard-truncated and the most recent boxed answer in y (if any) is taken as the prediction, otherwise the request is scored as incorrect. (ii) **s1** (Muennighoff et al., 2025): iterative budget forcing that suppresses end-of-turn with a WAIT cue until the budget is exhausted; no reduction direction. (iii) **TALE-EP** (Han et al., 2025): the budget B is stated explicitly in the prompt (e.g., "Please answer using less than B tokens for your reasoning."), and no further intervention is applied. This tests whether the model self-regulates when made aware of the budget.

Metrics. We report three metrics per (model, dataset, budget, method) cell: (i) *Accuracy* (%), exact-match on the boxed answer; (ii) *Token utilization* (%), defined as $100 \cdot \overline{|y|}/B$ where $\overline{|y|}$ is the mean response tokens per request (including all intervention cues for PARENTINGLLM); this measures how well a method exploits the available budget rather than the raw token count; (iii) *Burst rate* (%), the fraction of requests reaching B without a well-formed boxed answer.

5.2 Main Results

PARENTINGLLM improves the accuracy–budget trade-off across three Gemma-4 models and four reasoning benchmarks (Table 1; per-model maturity trajectories and representative cases in Fig. 4). It raises example-weighted average accuracy over Vanilla by 0.4–2.3 points and reduces burst rate by up to 3.4 points, achieving the best accuracy in 12 of 24 and the best burst rate in 15 of 24 model–benchmark–budget settings.

The strongest gains concentrate on **AIME 24+25**, the hardest benchmark. At $B = 2048$, PARENTINGLLM reduces average AIME burst from 50.0% to 39.4% across the three models and improves accuracy by 5.0 points over the best baseline. At $B = 1024$, where the budget is too tight for many AIME problems, burst is reduced from 78.9% (Vanilla) to 72.8%, outperforming TALE-EP's 75.5%; accuracy gains are modest (+0.6 over the best baseline), suggesting that small budgets impose a hard floor on achievable accuracy. HURRY interventions are most useful when maturity lags behind budget consumption (Fig. 4(d)).

TALE-EP confirms that exposing the budget in the prompt can help, but does not fully solve budgeted commitment control: on AIME 24+25 it reduces burst over Vanilla, yet PARENTINGLLM reduces it further, especially at $B = 2048$. The relative strength of the two methods varies with base-model capability: TALE-EP is competitive on Gemma-4-31B-it for MATH-500 and GPQA-Diamond, where sufficiently capable models self-regulate from the prompt alone, while PARENTINGLLM's gains concentrate on smaller models (E4B-it, A4B-it) and the hardest benchmark (AIME 24+25), where prompt-based self-regulation leaves substantial residual burst The two are complementary rather than competing. s1-style budget forcing mostly increases token utilization with only marginal accuracy gains, consistent with its one-sided design: it can extend reasoning but cannot apply finalization pressure when maturity lags. Fig. 4(a–c) shows this directly: OURS pulls the maturity trajectory toward the target schedule across all model scales, while Vanilla drifts above or below depending on the benchmark.

GSM-Hard as a held-out generalization test. GSM-Hard (Gao et al., 2023) also serves as a distribution shift in our setting, since no GSM-family data is used to fit f_θ or tune (τ_W, τ_H). PARENTINGLLM improves or matches Vanilla accuracy in

Table 1: Main results across three models, four benchmarks, and two budgets ($B \in \{1024, 2048\}$). For each cell, Acc denotes Accuracy (%), Util denotes Token Utilization (%, mean response tokens divided by B), and Burst denotes Burst Rate (%, fraction of requests reaching B without a well-formed boxed answer). Avg is the example-weighted average of each metric across the four benchmarks, with Δ relative to Vanilla shown in parentheses ($+$ improvement for Acc/Util, $-$ improvement for Burst).

B	Method	MATH-500			GPQA-Diamond			AIME 24+25			GSM-Hard			Avg		
		Acc.	Util.	Burst	Acc.	Util.	Burst	Acc.	Util.	Burst	Acc.	Util.	Burst	Acc. (Δ)	Util.	Burst (Δ)
Gemma-4-E4B-it																
1024	Vanilla	67.6	57.9	22.0	33.3	90.4	65.7	6.7	98.7	93.3	37.7	40.6	5.8	43.6	51.2	17.9
	s1	67.6	60.7	21.8	33.3	92.7	65.7	6.7	98.7	93.3	38.2	58.2	5.8	43.9 (+0.3)	63.3	17.9 ($-$0.0)
	TALE-EP	**69.0**	57.4	**21.4**	39.4	89.1	**62.6**	**10.0**	98.0	90.0	37.6	40.6	6.4	44.5 (+0.9)	63.3	17.9 ($-$0.0)
	OURS	67.2	63.4	22.4	**37.9**	89.4	63.6	6.7	97.4	**88.3**	**39.4**	51.6	6.1	**45.0 (+1.4)**	59.37	17.9 ($-$0.0)
2048	Vanilla	76.6	35.7	8.4	51.0	67.7	31.8	26.7	85.0	60.0	37.8	21.6	0.9	48.1	31.2	7.4
	s1	76.2	37.4	8.4	51.0	70.5	31.8	26.7	85.6	60.0	38.4	31.6	0.9	48.4 (+0.3)	38.3	7.4 ($-$0.0)
	TALE-EP	77.6	36.3	7.8	49.5	69.4	32.3	28.3	86.0	55.0	38.4	22.0	**0.5**	48.6 (+0.5)	31.8	6.9 ($-$0.5)
	OURS	**78.2**	40.5	**7.4**	**53.0**	66.6	**29.8**	**31.7**	84.2	**51.7**	**40.3**	30.1	0.7	**50.4 (+2.3)**	**37.7**	**6.6 ($-$0.8)**
Gemma-4-26B-A4B-it																
1024	Vanilla	68.2	55.6	16.8	53.5	86.7	55.6	25.0	93.9	73.3	61.3	43.8	13.2	61.2	52.2	19.8
	s1	67.2	55.7	16.8	53.5	86.7	55.6	25.0	93.9	73.3	**62.5**	54.7	13.2	61.7 (+0.5)	59.1	19.8 ($-$0.0)
	TALE-EP	69.6	54.9	15.2	55.6	84.8	43.9	25.0	92.7	73.3	60.9	54.7	**12.4**	61.5 (+0.3)	58.7	17.8 ($-$2.0)
	OURS	**71.4**	52.4	**12.8**	**63.1**	75.2	**30.8**	**30.0**	90.6	**68.3**	61.0	48.6	13.3	**62.8 (+1.6)**	53.3	**16.4 ($-$3.4)**
2048	Vanilla	76.4	33.0	5.8	71.7	60.4	21.2	46.7	78.0	51.7	62.5	27.0	9.2	66.3	33.1	10.8
	s1	74.6	33.1	5.8	71.7	60.4	21.2	46.7	78.0	51.7	**63.2**	33.8	9.2	66.3 (+0.0)	37.4	10.8 ($-$0.0)
	TALE-EP	75.8	33.2	5.2	68.7	63.6	26.3	46.7	77.9	51.7	61.9	28.4	9.6	65.5 (- 0.8)	34.3	11.4 ($+$0.6)
	OURS	**78.0**	33.8	**4.2**	71.7	52.4	**16.2**	**56.7**	74.0	**38.3**	62.6	34.6	9.6	**67.0 (+0.7)**	37.2	**9.8 ($-$1.0)**
Gemma-4-31B-it																
1024	Vanilla	75.0	49.1	8.6	68.7	72.3	25.8	28.3	91.7	70.0	59.4	33.6	4.2	63.1	42.7	9.2
	s1	74.8	49.2	8.6	68.7	72.3	25.8	28.3	91.7	70.0	**60.3**	42.6	4.2	63.7 (+0.6)	48.4	9.2 ($-$0.0)
	TALE-EP	**76.2**	47.9	**7.4**	**73.2**	68.4	18.2	33.3	89.6	63.3	59.7	33.0	**3.3**	**64.2 (+1.1)**	41.6	**7.4 ($-$1.8)**
	OURS	75.0	47.8	7.6	72.7	57.1	**13.1**	**33.3**	89.4	**61.7**	59.5	41.9	5.1	63.7 (+0.6)	46.1	8.1 ($-$1.1)
2048	Vanilla	80.0	26.2	1.6	77.8	41.3	3.5	55.0	70.0	38.3	59.7	18.2	2.0	66.2	23.8	3.1
	s1	80.2	26.3	1.6	77.8	41.3	3.5	56.7	70.2	38.3	60.0	22.9	2.0	66.5 (+0.3)	26.8	3.1 ($-$0.0)
	TALE-EP	**80.4**	26.5	**1.4**	77.8	41.4	7.1	58.3	67.4	33.3	59.8	18.6	2.0	66.4 (+0.2)	24.1	3.3 ($+$0.2)
	OURS	78.8	29.6	**1.4**	76.8	35.8	5.1	**60.0**	69.5	**28.3**	**60.8**	25.2	**1.9**	**66.6 (+0.4)**	28.6	**2.8 ($-$0.3)**

five of six model–budget settings, with the largest gains on Gemma-4-E4B-it, but burst-rate reductions are mixed (Fig. 4(e) shows a representative underthink case). This weakens the explanation that the gains stem purely from benchmark-specific calibration, while highlighting a limitation: out-of-domain arithmetic traces do not always benefit from the same burst-reduction behavior.

Table 2: Signal ablation on Gemma-4-E4B-it: alternative inputs to the maturity estimator f_θ.

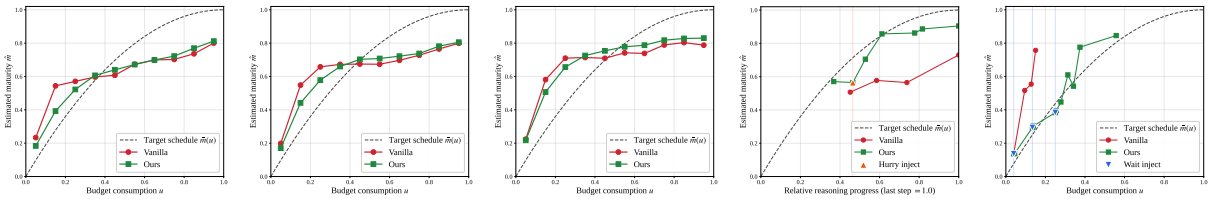
f_θ inputs	$B = 1024$		$B = 2048$	
	Acc.	Burst	Acc.	Burst
Vanilla	43.6	17.9	48.1	7.4
u_i only	44.8 ($+$1.2)	18.7 ($+$0.8)	49.7 ($+$1.6)	6.9 ($-$0.5)
u_i + entropy	44.7 ($+$1.1)	18.7 ($+$0.8)	49.1 ($+$1.0)	7.2 ($-$0.2)
$u_i + P_{\text{EOT}}$	**45.0 ($+$1.4)**	17.9 ($-$0.0)	**50.4 ($+$2.3)**	**6.6 ($-$0.8)**

5.3 Ablation Studies

We isolate the contribution of each design choice through two ablations on Gemma-4-E4B-it across all four benchmarks.

A1. Signal ablation. We replace the input to f_θ with alternative signals (Table 2): u_i alone (budget consumption only), u_i paired with next-token entropy, and the full (u_i, P_{EOT}). The full configuration outperforms both alternatives. Using u_i alone is blind to whether the prefix is actually ready to finalize, while replacing P_{EOT} with entropy degrades performance, consistent with Fig. 3: entropy is non-monotonic in reasoning progress and therefore cannot serve as a maturity proxy.

A2. Direction ablation. We isolate the value of bidirectional control by comparing single-direction variants (Table 3): *Wait-only* (no HURRY), *Hurry-only* (no WAIT), and the full bidirectional controller. The bidirectional version is the only setting that



(a) Gemma-4-E4B (b) Gemma-4-26B-A4B (c) Gemma-4-31B (d) Case : Overthink (e) Case : Underthink

Figure 4: Schedule tracking and representative cases at $B = 2048$. **(a–c)** Per-model average maturity trajectories: estimated maturity $\hat{m}$ averaged across all evaluation traces. Vanilla (red) is the model's natural trajectory; OURS (green) adds the maturity-gap controller. The black dashed line is the target schedule $\bar{m}(u) = 1 - (1 - u)^2$; OURS tracks $\bar{m}$ more closely than Vanilla across all model scales. **(d)** A representative best-case sample where Vanilla overthinks: maturity lags the schedule and the budget is exhausted before finalization; the controller injects HURRY to apply finalization pressure. **(e)** A representative best-case sample where Vanilla underthinks: maturity exceeds the schedule and the model commits prematurely; the controller injects WAIT to delay commitment.

Table 3: Direction ablation on Gemma-4-E4B-it: bidirectional control vs. WAIT and HURRY-only variants.

Action set	$B = 1024$		$B = 2048$	
	Acc.	Burst	Acc.	Burst
Vanilla	43.6	17.9	48.1	7.4
w/ Wait	34.3 (-9.3)	26.7 ($+8.8$)	48.7 ($+0.6$)	8.3 ($+0.9$)
w/ Hurry	**45.9** (**$+2.3$**)	**15.3** (**-2.6**)	48.8 ($+0.7$)	**6.2** (**-1.2**)
OURS	45.0 ($+1.4$)	17.9 (-0.0)	**50.4** (**$+2.3$**)	6.6 (-0.8)

Table 4: Probe overhead on Gemma-4-E4B-it at $B = 2048$, per-request means. N_p: probes per request. T_p: total probe tokens. T_m: main-path tokens for OURS and Vanilla. Net Δ: total forward-pass tokens vs. Vanilla.

Dataset	N_p	T_p	T_m^{OURS}	$T_m^{\text{Van.}}$	Net Δ
AIME 24+25	15.0	438	1725	1867	$+15.9\%$
MATH-500	11.0	243	829	900	$+19.0\%$
GPQA-D	12.0	403	1363	1580	$+11.8\%$
GSM-Hard	10.6	240	617	442	$+93.9\%$

improves accuracy across both budgets. Wait-only fails under tight budgets ($B = 1024$): without finalization pressure, additional verification cues push more requests past the budget, raising burst rate. Hurry-only is competitive at $B = 1024$ but underperforms at $B = 2048$, where premature commitment is not corrected on problems that would benefit from extended reasoning. The bidirectional rule resolves both failure modes by selecting the intervention direction from the maturity-gap sign.

5.4 Probe Overhead

Table 4 reports probe cost on Gemma-4-E4B-it at $B = 2048$. Each probe is short (22–34 tokens) because the commit suffix elicits only a boxed answer. On AIME 24+25, MATH-500, and GPQA-D, probe cost is partially offset by main-path compression from HURRY cues, yielding a net overhead of $+11.8\%$ to $+19.0\%$. On GSM-Hard, Vanilla traces are short (442 tokens) and the controller injects WAIT more often than HURRY, prolonging the main path; the $+93.9\%$ net cost reflects intervention against premature commitment rather than probe overhead. KV-cache impact is bounded by one probe-path length per concurrent request.

6 Limitations and Future Work

Single-family evaluation. Our experiments span three Gemma-4 models at different parameter scales but share a single instruction-tuning recipe and chain-of-thought format. Validating PARENTINGLLM on other reasoning model families (e.g., Qwen3.5, DeepSeek-R1), including those that emit reasoning without explicit step markers, is the most important next step.

Probe overhead and supervision. The probe runs on a forked path at every boundary, adding KV-cache and forward-pass cost; we have not measured this under serving load. The maturity estimator also inherits any bias from the LLM judge used to label its training data. Approximating P_{EOT} without an explicit probe, and replacing judge labels with task-grounded signals, are open directions.

Maturity label dependency. Training labels for f_θ come from an LLM judge whose monotone output is enforced by clamping (Appendix B.2); the estimator therefore inherits the judge's biases, and the monotonicity of $\hat{m}$ along u is partly an artefact of the label-shaping procedure. Replacing judge supervision with a task-grounded target (e.g., final-answer correctness on held-out samples) is a natural next step.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, and 1 others. 2025. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and 1 others. 2024. Efficiently scaling llm reasoning with certaindex. *arXiv preprint arXiv:2412.20993*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International conference on machine learning*, pages 10764–10799. PMLR.

Google. 2026. Gemma 4. https://huggingface.co/collections/google/gemma-4. Hugging Face collection.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2026. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31176–31184.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. 2025. Steering llm thinking with budget guidance. *arXiv preprint arXiv:2506.13752*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

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

OpenAI. 2025. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive

slow-thinking for large reasoning models. *URL https://arxiv. org/abs/2503.04472*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Xi Wang, Lequn Wang, James McInerney, and Nathan Kallus. 2026. Eat: Entropy after </think> for reasoning model early exiting.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

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

Hang Yan, Fangzhi Xu, Rongman Xu, Yifei Li, Jian Zhang, Haoran Luo, Xiaobao Wu, Luu Anh Tuan, Haiteng Zhao, Qika Lin, and 1 others. 2025. Mur: Momentum uncertainty guided reasoning for large language models. *arXiv preprint arXiv:2507.14958*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX symposium on operating systems design and implementation (OSDI 22)*, pages 521–538.

Xichen Zhang, Sitong Wu, Haoru Tan, Shaozuo Yu, Yinghao Zhu, Ziyi He, and Jiaya Jia. 2025. Smartswitch: Advancing llm reasoning by overcoming underthinking via promoting deeper thought exploration. *arXiv preprint arXiv:2510.19767*.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210.

A Implementation Details

A.1 Experimental Environment

All experiments are run on a single node with 8×NVIDIA H200 GPUs (141 GB HBM3e each), CUDA 12.8, and PyTorch 2.10. Inference uses our modified vLLM runtime, built on the public vLLM v0.19.0 codebase, with continuous batching enabled and tensor-parallel size 1 for Gemma-4-E4B-it, 2 for Gemma-4-26B-A4B-it, and 4 for Gemma-4-31B-it. The maturity estimator f_θ is trained with LightGBM 4.6.0 on a separate CPU node. We use vLLM sampling parameters `temperature=0.0`, `top_p=1.0` identical across all methods and baselines to isolate the contribution of the controller.

A.2 Reasoning Boundary Definition

A reasoning boundary is a paragraph-level segmentation point in the generated trace at which the controller is invoked. Concretely, the sampler fires a boundary when (i) the last emitted token terminates the current paragraph(either a single token whose decoding ends in `\n\n`, or two consecutive single-newline tokens) and (ii) the next token's `argmax` falls within the set of `Step`-prefixed tokens (e.g., `Step`, `Step␣1`, `Step:`). This pattern matches the explicit `Step␣1:␣...␣Step␣2:␣...` chain-of-thought format that the instruction-tuned Gemma-4 models follow by default under our task prompt template. The set of `\n\n`-ending tokens and `Step`-prefix tokens is derived per model at engine startup by scanning the tokenizer vocabulary, so the same definition transfers to other instruction-tuned models without changes; this vocabulary scan is performed once per model load and adds no runtime overhead during inference.

Empirically, at least one boundary fires on $\geq 99.8\%$ of reasoning traces across all five evaluation datasets (AIME 24+25, MATH-500, GPQA-Diamond, GSM-Hard) and all three Gemma-4 model sizes.

A.3 Wait and Hurry Cue Strings

Both cues are short natural-language insertions appended directly to the main decoding path. The exact strings used in all experiments are framed as out-of-band user interjections (mimicking a user breaking into the assistant's mid-response), since instruction-tuned models react more strongly to a perceived user message than to a self-uttered hedge:

- WAIT cue: "`(user: wait, check your answer)`"
- HURRY cue: "`(user: hurry up and finalize)`"

The user-interjection framing was selected after preliminary experiments showed that politer hedges in the assistant's own voice (e.g., "`Based on the analysis so far, the answer is`") produced weaker termination behavior on AIME 24+25, where the model otherwise tends to over-elaborate. We also observed that more urgent register strengthens the termination effect on the largest model (Gemma-4-31B-it); the strings released in our reference implementation include this more direct register, and the version above is provided for reproducibility under a sanitized formulation. Both strings are tokenized by the model's native tokenizer and count toward the response budget B.

A.4 Maturity Estimator Training

The maturity estimator f_θ is a LightGBM (Ke et al., 2017) gradient-boosted regressor mapping $(u_i, P_{\text{EOT},i}) \in [0,1]^2$ to a scalar maturity score in $[0,1]$. Hyperparameters (Table 5) are shared across all three Gemma-4 sizes; we did not tune them per model.

Training data. For each base model, we generate *probe-only* boundary traces(the controller's PROGRESS_MEASURE mode, which inserts the probe at every boundary but performs no WAIT/HURRY injection) on the calibration splits of MATH-500, GPQA-Diamond, and AIME 24+25 at $B = 2048$. Each boundary yields one $(u_i, P_{\text{EOT},i})$ feature pair and a judge-derived maturity label (Appendix B.2). GSM-Hard is held out and used only as an out-of-distribution evaluation set.

Judge model. Labels come from an open-weight judge, `openai/gpt-oss-120b`, served locally on the same vLLM cluster. The judge outputs a monotone non-decreasing cumulative progress array (Appendix B.2); scores are clipped to $[0,1]$ and used directly as the regression target. The judge is invoked only at training time and never at inference.

A.5 Baseline Implementations

All baselines use the same sampling parameters as PARENTINGLLM and run inside the same vLLM runtime to isolate the contribution of the controller.

Table 5: LightGBM hyperparameters for f_θ, shared across all base models.

Parameter	Value
`num_boost_round`	500 (early stop 30)
`learning_rate`	0.05
`num_leaves`	31
`min_data_in_leaf`	50
`feature_fraction`	0.9
`bagging_fraction`	0.9
`bagging_freq`	5
Objective	regression (RMSE)

Vanilla. The model decodes freely until either it emits the end-of-turn token or $|y| = B$. If hard-truncated, we attempt to extract the most recent well-formed `\boxed{...}` answer from y; if none exists, the request is scored as incorrect.

s1. A client-side re-implementation of budget forcing (Muennighoff et al., 2025). We iteratively suppress termination: whenever the model emits the end-of-turn token (`finish_reason="stop"`) with budget remaining, we append the literal cue "`␣Wait`" and re-issue the request, conditioning on the accumulated assistant text. This repeats until either the model fails to emit a boxed answer before exhausting B, or no budget remains. If the final continuation overruns B mid-generation, we recover the last brace-balanced `\boxed{...}` from the prior natural-stop turn rather than the truncated tail. Cues are injected only on natural stop, never after length truncation.

TALE-EP. Following Han et al. (2025), the prompt is augmented with the explicit instruction "`Please answer using less than` B `tokens for your reasoning.`", where B is the actual budget. No further inference-time intervention is applied. This isolates whether the model self-regulates when made aware of the budget.

B Prompts

B.1 Task Prompt Templates

The user message concatenates a domain-specific instruction, the problem, and the `META_DUAL_EXEMPLAR` control-signal block. This block is shared across all methods (including vanilla, s1, and TALE-EP) for prompt parity; signals are injected only by PARENTINGLLM.

Mathematics (MATH-500, AIME 24+25, GSM-Hard).

```
Solve the math problem step by step.
Format each reasoning step on its own
with a 'Step N:' header. Separate steps
with a blank line so each new step starts
with "\n\nStep N:". After all reasoning
steps, on a new paragraph write exactly
'The Final Answer is \boxed{<answer>}.'

Problem: {problem}

Begin your solution with 'Step 1:'.

{META_DUAL_EXEMPLAR}
```

GPQA-Diamond. Four options are deterministically permuted using SHA-256 of the record ID and pre-concatenated as "A) , B) , C) , D) ".

```
You are an expert answering hard
graduate-level science multiple choice
questions. Solve the question step by
step. Format each reasoning step on
its own with a 'Step N:' header. ...
After all reasoning steps, write exactly
'The Final Answer is \boxed{<single
letter>}.' where <single letter> is A,
B, C, or D.

Question (choose one letter):
{problem with embedded "A) ...\n B) ..."}

Begin your solution with 'Step 1:'.

{META_DUAL_EXEMPLAR}
```

{META_DUAL_EXEMPLAR}

```
**Important control signals** (the
SYSTEM may insert one of these phrases
between your reasoning steps as an
interruption from the user side):

- '(user: hurry up and finalize)' — the
user is impatient. Briefly acknowledge,
then continue with much shorter, more
clipped phrasing. Drop filler and
self-explanation.
- '(user: wait, check your answer)'
— the user wants you to double-check
the previous step. Briefly acknowledge
("Let me reconsider"), then revisit
the most recent step with a careful
re-derivation.

Example (both signals on the same
problem):
Problem: Find the roots of x^2 - 5x + 6
= 0.
Step 1: Discriminant 25 - 24 = 1, perfect
square; roots are integers.
Step 2: Factors of 6 summing to -5: (-2,
-3).
(user: wait, check your answer)
Step 2 (re-check): (x-2)(x-3) = x^2 - 5x
+ 6. Matches.
Step 3: From (x-2)(x-3) = 0, one factor
is zero.
(user: hurry up and finalize)
Step 4: x = 2 or x = 3.
The Final Answer is \boxed{2, 3}.

CRITICAL: Do NOT generate these (user:
...) phrases yourself.
```

The two phrases are the exact strings injected by PARENTINGLLM's controller at \n\n boundaries.

B.2 Judge Prompt for Maturity Labels

Training labels for f_θ come from an external judge (openai/gpt-oss-120b) that scores all reasoning steps of a single trace jointly as a monotone non-decreasing cumulative progress curve. The trace is split on blank-line boundaries (regex \n\s*\n) into N steps; each step is truncated to 800 characters with a "...[trunc]" marker when needed. The judge sees the ground-truth final answer (target) to ground its progress estimates; this is acceptable because labels are used only for training of f_θ.

```
You are evaluating a reasoning trace.
CORRECT FINAL ANSWER: {target}
The trace below is split into {N}
numbered steps separated by blank lines.
For EACH step, output the CUMULATIVE
progress fraction toward the final
answer at the end of that step, as a
decimal in [0.00, 1.00].

Guidelines:
0.00–0.15: just started; 0.15–0.40:
approach chosen, derivation begun;
0.40–0.70: main reasoning; 0.70–0.95:
approaching answer; 0.95–1.00: final
answer committed (boxed).

The values must be MONOTONE
NON-DECREASING. Output STRICTLY a
JSON array of EXACTLY {N} floats in
order. NOTHING ELSE.
STEPS: [1] {step_1_text} [2]
{step_2_text} ... [N] {step_N_text}
```

Non-monotone outputs are clamped post-hoc by running maximum. Step k's label is paired with the budget consumption u_k at the matching \n\n boundary.

B.3 Forced-Commit Probe Suffix

At every \n\n boundary, the probe path forks the main KV cache and appends the literal suffix

```
" The Final Answer is"
```

(leading space; re-tokenised per tokenizer at engine startup for cross-model portability). The probe freely generates up to 64 tokens with logprobs $=$ 20; a brace-balanced parser locates the proper closing } of `\boxed{...}`. The probe terminates—and its KV blocks are freed—once five positions are emitted beyond the closing brace; $P_{\mathrm{EOT},i}$ is the maximum end-of-turn probability over those five trailing positions.