BRIDGING THE GAP BETWEEN LATENT AND EXPLICIT REASONING WITH LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models typically reason via explicit chain-of-thought (CoT), generating intermediate steps token-by-token. Latent CoT offers an alternative: it performs multi-step reasoning in the model's hidden states, replacing decoded tokens with continuous representations for greater efficiency. However, existing latent CoT methods underperform explicit CoT beyond 1B parameters, and the gap widens with scale. *Looped*, or recurrent-depth, Transformers, which reuse their weights to increase computation depth without adding parameters, are a natural fit for latent reasoning. We therefore ask whether looped Transformers can bridge this gap. We answer affirmatively with a simple recipe: a *looped padded Transformer* that processes K latent blocks in parallel for R iterations, with a cross-entropy loss on each latent position's gold CoT-step token, similar to explicit CoT supervision. We instantiate it as **LOTUS** (Looped Transformers with parallel supervision on latents). LOTUS is, to our knowledge, the first latent-CoT method to bridge the gap to explicit CoT at the 3B scale, while cutting thought-phase latency by $2.5\times$–$6.9\times$ from compact math expressions to natural language. Projecting LOTUS's post-loop latents through the base LM head recovers the gold reasoning steps and even surfaces alternative valid intermediate steps, evidence that its latent space is interpretable and CoT-aligned. Ablations confirm that both the looped backbone and the parallel supervision on gold CoT tokens are essential.

1 INTRODUCTION

Scaling inference compute, i.e., letting a model "think" before it answers, has become a dominant lever for increasing language model capabilities, with stronger performance now coming from longer reasoning chains rather than from model size alone (DeepSeek-AI, 2025; OpenAI, 2026). Chain-of-thought (CoT) reasoning (Wei et al., 2022), where the model emits intermediate reasoning steps, drives this trend. However, since each token must be decoded sequentially, generating a CoT of length N takes N sequential model evaluations, making reasoning costly.

Latent reasoning aims to achieve the same at a fraction of the cost: it carries out the intermediate computation in continuous hidden states rather than decoded tokens, condensing many steps into fewer model evaluations. On small backbones such as GPT-2 (Radford et al., 2019), latent methods (Hao et al., 2025; Shen et al., 2025; Wei et al., 2025) already match CoT accuracy. Yet at the scales where CoT matters most, the promise breaks down: beyond 1B parameters, no existing latent method matches explicit CoT on math reasoning, and the gap widens with model size (Wei et al., 2025).

We identify two common issues in latent methods. **(P1)** Sequential generation: methods like Coconut (Hao et al., 2025), CODI (Shen et al., 2025), and SIM-CoT (Wei et al., 2025) produce latent thoughts in a pure autoregressive way, so the number of sequential forward passes still scales linearly with the number of latent tokens, keeping CoT's sequential generation bottleneck while giving up the readable intermediate steps that explicit CoT provides. **(P2)** Lack of CoT grounding: Explicit CoT gives every reasoning step a direct, position-aligned token target. Without similarly grounded supervision, latent traces can drift from meaningful computation and destabilize training at scale (Li et al., 2026b; Zou et al., 2026). Methods like PCCoT (Wu et al., 2025) and KaVa (Kuzina et al., 2026) mitigate the sequential bottleneck with Jacobi-style iteration but still ground their latents through

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

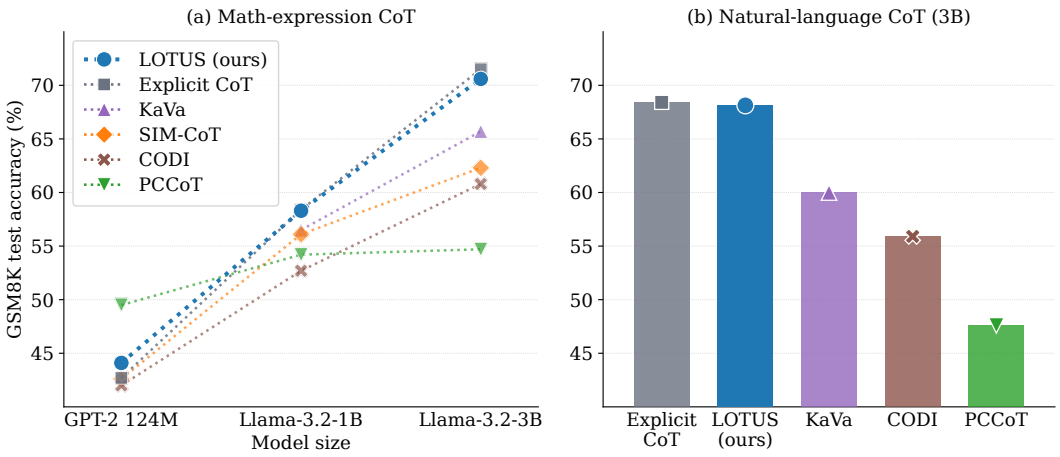


Figure 1: LOTUS bridges the latent–explicit CoT accuracy gap across scale on GSM8K test. **(a)** Math-expression CoT across model sizes: prior latent methods fall further below explicit CoT as the backbone grows, whereas LOTUS tracks the explicit-CoT ceiling while cutting thought-phase latency by $2.5\times$ at 3B. **(b)** Natural-language CoT at 3B: LOTUS matches explicit CoT and outperforms the latent baselines while reducing thought-phase latency by $6.9\times$.

indirect signals that fall short of CoT-aligned target, such as hidden-state distillation from a teacher CoT model or a compressed teacher key–value cache.

An architecture that resolves both issues at once would refine its latents in a few parallel passes rather than one autoregressive step per token, and would ground them in a target as direct as explicit CoT's. *Looped*, or recurrent-depth, Transformers are a natural fit for the first requirement: they reuse the same weights across iterations to add computation depth without extra parameters, a design that recent work scales to billion-parameter pretraining (Geiping et al., 2025a; Zhu et al., 2025c; Zeng et al., 2026b). On parallelizable problems, they can provably reach a solution in fewer model iterations CoT (Saunshi et al., 2025); for example, LPTs solve graph reachability in a logarithmic number of iterations (Merrill & Sabharwal, 2025b)—an exponential improvement over sequential latent methods. This raises the question we study: *can a looped Transformer bridge the gap to explicit CoT while reasoning in fewer sequential steps?*

We answer with **LOTUS** (Looped Transformers with parallel supervision on latents). It refines K padded latent blocks of c tokens over $R \ll N$ parallel iterations, addressing **P1**, and supervises each latent position against its gold CoT-step token through the base LM head, addressing **P2**. We also study LOTUS-aux, which routes the same supervision through an auxiliary decoder under teacher forcing. Our contributions are:

- A simple latent-reasoning recipe combining padded looped computation and parallel gold-token supervision (Section 3), with direct and auxiliary-decoder routings.
- Near-explicit-CoT accuracy at 3B (Figure 1 and Section 4.2): LOTUS is within 1.5 points on GSM8K, or within 1 point with CODI, and surpasses the explicit-CoT out-of-domain average. Thought-phase speedups are $2.5\times$ for math-expression and $6.9\times$ for natural-language CoT.
- Ablations establishing the importance of parallel supervision, block width, and loop depth (Section 4.3). Direct readout is robust across scale; the auxiliary routing matches it at 3B but degrades on smaller backbones.
- Readable latent representations that recover gold tokens, support unseen valid alternatives, and reflect complementary step and answer losses (Appendix A).

2 Preliminaries

A glossary of all notation is provided in Table 11 in Appendix E.

Explicit CoT. A **language model** (LM) $f_{\boldsymbol{\theta}}$ can map a question Q to an answer A via multi-step reasoning: generating S intermediate steps $T_1, \ldots, T_S$ before the answer, where T_i ($i \in \{1, \ldots, S\}$) may span multiple tokens. We write N for the total number of CoT tokens across all S steps. Standard CoT supervision trains all CoT tokens in parallel via teacher forcing, but at inference time the tokens must be generated sequentially, increasing latency. We refer to this setup as *Explicit* CoT.

Latent reasoning replaces decoded CoT tokens with multi-step inference in the model's hidden states, which are commonly referred to as latent thoughts. Zhu et al. (2025a) show latent thoughts carry strictly more information than discrete tokens: linearly many latent steps in a Transformer can solve graph reachability that would otherwise require quadratically many explicit CoT steps.

Looped padded Transformers. A **looped Transformer** (Dehghani et al., 2019) applies $f_{\boldsymbol{\theta}}$'s backbone multiple times, gaining depth without adding parameters. *Padding* the input with extra learnable tokens (Goyal et al., 2024; Pfau et al., 2024) adds parallel computation to each iteration. The combination of the two creates **looped padded Transformers** (LPTs), which are more capable than either component alone.

Prior latent reasoning methods. We contrast LOTUS against the most closely related latent reasoning methods, which we summarize here (details in Appendix F). *Coconut* (Hao et al., 2025) replaces each CoT step with a c latent tokens generated autoregressively from the previous one, under a curriculum, and supervises only the answer. CODI (Shen et al., 2025) keeps this autoregressive latent decoding but adds a distillation target, aligning one designated latent's hidden state with the corresponding hidden state of a teacher CoT model. SIM-CoT (Wei et al., 2025) instead trains an auxiliary autoregressive decoder that aligns each latent with its CoT token. PCCoT (Wu et al., 2025) and KaVa (Kuzina et al., 2026) drop the autoregressive nature, refining a fixed budget of latent tokens in parallel over a few Jacobi iterations; they ground the latents indirectly, through CODI-style distillation and a compressed teacher key–value cache, respectively. We refer the reader to Appendix D for an extended discussion of related work on latent reasoning and looped Transformers, and to Appendix F for a detailed comparison with the related latent reasoning methods.

3 METHOD

Although LPTs are expressive enough to carry out efficient parallel reasoning, without structured supervision they often fail to generalize out-of-distribution (Altabaa et al., 2025). This motivates our central design choice: grounding the latent computation directly in the gold CoT tokens. We turn a standard LM into a latent reasoner with a recipe built from two ingredients: *(i)* a *padded latent prefix processed by an LPT*, and *(ii) parallel cross-entropy supervision on exact gold CoT tokens*. We instantiate this recipe as **LOTUS** (Looped Transformers with parallel supervision on latents, Figure 2), which reads the supervision directly through the base model's LM head. We also study LOTUS-aux (Figure 3), which instead routes the supervision through an auxiliary decoder. We detail the padded latent prefix in Section 3.1, the looped computation in Section 3.2, and LOTUS's objective in Section 3.3. Supporting loss-factorization analysis and the full LOTUS-aux formulation are provided in Appendices B and C.

3.1 PADDED LATENT PREFIX

For a question Q, S chain-of-thought (CoT) steps $T_1, \ldots, T_S$, and answer $A = (A_1, \ldots, A_{|A|})$ (with $A_{|A|} = \langle \text{EoS} \rangle$), we construct an input

$$\boldsymbol{x} = \big[Q,\ \langle \texttt{BoT} \rangle,\ \underbrace{\langle \texttt{lat} \rangle \cdots \langle \texttt{lat} \rangle}_{\text{block 1 } (c \text{ tokens})},\ \cdots,\ \underbrace{\langle \texttt{lat} \rangle \cdots \langle \texttt{lat} \rangle}_{\text{block } K \ (c \text{ tokens})},\ \langle \texttt{EoT} \rangle,\ A \big], \tag{1}$$

where $\langle \texttt{lat} \rangle$ is a learnable latent token shared across positions, and $\langle \texttt{BoT} \rangle, \langle \texttt{EoT} \rangle$ are learnable special tokens that delimit the latent region. The block budget K and per-block width c are fixed hyperparameters, so the latent region contains Kc latent tokens plus the two delimiters. The latent prefix follows prior work on padding or pause tokens (Goyal et al., 2024; Pfau et al., 2024), with each block i intended to encode the latent representation of CoT step T_i, $i \in \{1, \ldots, K\}$. Unlike the example-dependent step count S of Section 2, the block budget K is fixed across examples; we

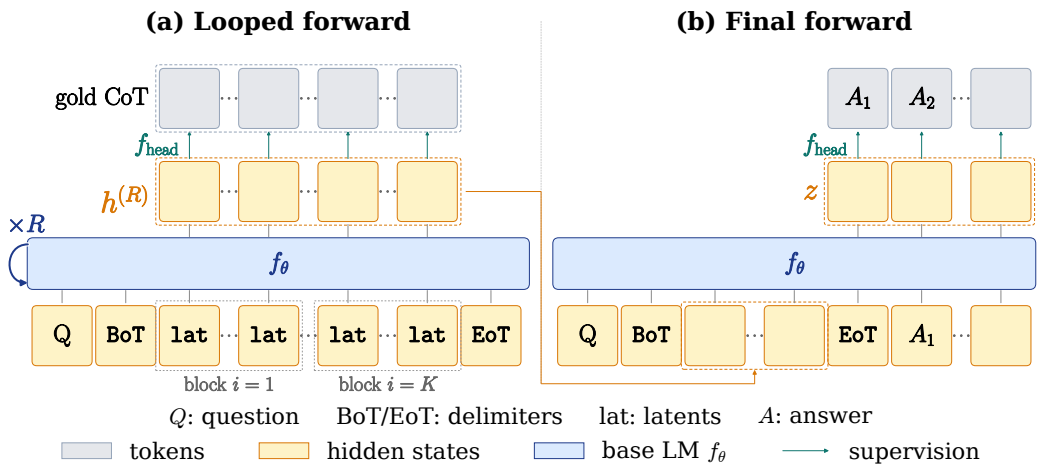


Figure 2: LOTUS architecture. (a) Looped forward: the looped LM $f_{\boldsymbol{\theta}}$ is iterated R times over $[\langle\texttt{BoT}\rangle, \langle\texttt{lat}\rangle \cdots \langle\texttt{lat}\rangle, \langle\texttt{EoT}\rangle]$, producing post-loop hidden states $\boldsymbol{h}^{(R)}$ at the latent positions, and $\mathcal{L}_{\text{step}}$ supervises these states through the LM head, in parallel, against the corresponding CoT tokens $T_{i,j}$. (b) Final forward: the post-loop latents $\boldsymbol{h}^{(R)}$ are inserted at the latent positions and the answer suffix A is supervised by next-token prediction ($\mathcal{L}_{\text{ans}}$) through the LM head.

choose it to cover the step counts in our data, so that $K \geqslant S$ on almost all examples and each block can align to one CoT step (Appendix G).

3.2 Looped Latent Computation

Let $f_{\boldsymbol{\theta}}$ be the base causal LM and $\boldsymbol{E} \in \mathbb{R}^{Kc \times d}$ the learnable latent embeddings, indexed as $\boldsymbol{E}_{i,j}$ for block $i \in \{1, \ldots, K\}$ and intra-block position $j \in \{1, \ldots, c\}$. LOTUS first runs a single forward pass over the prefix $[Q, \langle\texttt{BoT}\rangle]$ to populate a KV cache $\mathcal{C}_{\text{pre}}$, which is reused throughout training and inference without recomputation. It then iterates $f_{\boldsymbol{\theta}}$ for R iterations, each refining the latent embeddings while attending to $\mathcal{C}_{\text{pre}}$ (the trailing $\langle\texttt{EoT}\rangle$ and the answer suffix A do not enter the loop). Writing $\boldsymbol{h}^{(t)} \in \mathbb{R}^{Kc \times d}$ for the hidden states at the latent positions after iteration t, with entries $\boldsymbol{h}^{(t)}_{i,j}$:

$$\begin{aligned} \boldsymbol{h}^{(0)} &= f_{\boldsymbol{\theta}}(\boldsymbol{E} \mid \mathcal{C}_{\text{pre}}), \\ \boldsymbol{h}^{(t)} &= f_{\boldsymbol{\theta}}\Big(\boldsymbol{E} + \boldsymbol{h}^{(t-1)} \,\Big|\, \mathcal{C}_{\text{pre}}\Big), \quad t = 1, \ldots, R. \end{aligned} \tag{2}$$

where the brackets denote subsequence concatenation. This is a finite-unroll, input-injected recurrence over a looped Transformer (Dehghani et al., 2019; Fan et al., 2025), with a fixed unroll depth R and gradients propagated through all R iterations.

3.3 Direct Step-Aligned Supervision

3.3.1 Objective

Step CoT supervision loss. After R iterations of looping (Section 3.2), we supervise each latent position (i, j) in the grid with the corresponding CoT step token. Let $T_i = (T_{i,1}, \ldots, T_{i,c})$ be the tokenized CoT step i padded or truncated to c tokens. We apply a single batched cross-entropy that directly aligns each block to its target step:

$$\mathcal{L}_{\text{step}} = \frac{1}{N_{\text{step}}} \sum_{i=1}^{K} \sum_{j=1}^{c} \text{CE}\Big(f_{\text{head}}(\boldsymbol{h}^{(R)}_{i,j}),\, T_{i,j}\Big), \tag{3}$$

where f_{head} is the LM head and $N_{\text{step}} = \sum_i |T_i|$ is the total number of supervised (non-padding) CoT tokens.[1]

Answer supervision loss. $\mathcal{L}_{\text{ans}}$ is computed in a separate *final forward* of $f_{\boldsymbol{\theta}}$ that reuses the prefix cache $\mathcal{C}_{\text{pre}}$ and inserts the post-loop latent hidden states $\boldsymbol{h}^{(R)}$ at the latent rows:

$$\boldsymbol{z} \;=\; f_{\boldsymbol{\theta}}\Big([\langle \texttt{EoT} \rangle,\, A] \;\Big|\; [\mathcal{C}_{\text{pre}},\, \boldsymbol{h}^{(R)}]\Big)\,, \tag{4}$$

where $\boldsymbol{z}_m \in \mathbb{R}^d$ collects the resulting hidden state at each answer-suffix position $m \in \{0, 1, \ldots, |A| - 1\}$ ($m = 0$ being the trailing $\langle \texttt{EoT} \rangle$ position, which predicts A_1). We denote hidden states here by $\boldsymbol{z}$, distinct from the looped forward latents $\boldsymbol{h}^{(t)}$, to mark that they come from different sources. Applying the LM head f_{head}, standard next-token cross-entropy on the answer suffix gives:

$$\mathcal{L}_{\text{ans}} \;=\; \frac{1}{|A|} \sum_{m=0}^{|A|-1} \text{CE}(f_{\text{head}}(\boldsymbol{z}_m),\, A_{m+1})\,. \tag{5}$$

Given a step supervision weight λ_{step}, the full objective is

$$\mathcal{L} \;=\; \mathcal{L}_{\text{ans}} + \lambda_{\text{step}}\, \mathcal{L}_{\text{step}}. \tag{6}$$

Key properties of the objective. The supervision is *direct* through the answer-generating LM head, *parallel* across all K blocks, and *post-loop* rather than per-iteration. We ablate these choices in Section 4.3 and compare prior methods in Appendix F.

Inference. We cache the question once, perform R looped refinements, and decode only the answer suffix autoregressively through the base LM head, conditioned on $\boldsymbol{h}^{(R)}$ (Figure 2). Latent tokens are projected to token space only for training supervision or analysis, never for answer generation. This removes sequential thought-token decoding while preserving answer generation from the latent hidden states.

Complementary roles of the losses. The step loss supervises a parallel chain likelihood, grounding each latent position in the gold token distribution, while the answer loss encourages jointly useful latent states. The latent states themselves are computed jointly by the looped Transformer, despite the factorized readout objective. The full factorization and support-coverage argument are in Appendix B, with empirical loss ablations in Appendix A.3.

Auxiliary decoder variant. We also study LOTUS-aux, which preserves the looped padded backbone but routes step supervision through an auxiliary autoregressive decoder under teacher forcing rather than directly through the base LM head. The auxiliary decoder supervises a block at each loop iteration and is discarded at inference; answer generation is unchanged. Its full formulation, objective, and architecture are provided in Appendix C.

4 Experiments

We evaluate accuracy, inference efficiency, and the contribution of each design choice. Setup and main results follow; Section 4.3 studies supervision, width, depth, and inference-time adaptation. Latent representation analysis and full training and evaluation details are in Appendices A and G.

4.1 Setup

Datasets and backbones. Following SIM-CoT (Wei et al., 2025), we train on GSM8k-Aug (Deng et al., 2023) with 385k training samples and report in-domain accuracy on the GSM8K test set (Cobbe et al., 2021), together with three out-of-domain benchmarks: GSM-Hard, MultiArith, and SVAMP. We additionally train on a natural-language version of GSM8K-Aug (Wu et al., 2025) as an efficiency stress test, where each reasoning step is a full-sentence rationale rather than a compact math expression. We use three backbones in increasing size: GPT-2 (124M) (Radford et al., 2019), LLAMA-3.2-1B-INSTRUCT, and LLAMA-3.2-3B-INSTRUCT (Grattafiori et al., 2024).

[1] Padding tokens used to pad CoT steps to length c are ignored by the cross-entropy (and excluded from N_{step}).

Methods. We report **LOTUS** (K=6 blocks of c=25 tokens on the LLAMA backbones, c=13 on GPT-2, and R=6 looped iterations)[2] and **LOTUS + CODI**, which adds CODI's single-position distillation loss. The auxiliary decoder variants LOTUS-aux and LOTUS-aux + CODI (Appendix C) share the same configuration. We compare with **Explicit CoT** and **No-CoT** as reference, and with **CODI** (Shen et al., 2025) and **CODI + SIM-CoT** (Wei et al., 2025) for each backbone, which share the sequential latent reasoning budget (R=6) with LOTUS and differ only in the number of latent tokens per step. Latent methods that use a different sequential compute budget: Coconut (Hao et al., 2025), Coconut + SIM-CoT (with 10 sequentially generated latent thought in total), and the parallel-latent methods PCCoT (Wu et al., 2025) and KaVa (Kuzina et al., 2026) (with 3 sequential steps and 24 latent tokens in total) are compared separately in Table 12 and Table 13 (Appendix F).

4.2 MAIN RESULTS

LOTUS matches explicit CoT at scale, where prior latent methods fall behind. LOTUS scales well from GPT-2 to LLAMA-3.2-3B-INSTRUCT (Table 1). Prior latent methods match explicit CoT at small scale but fall behind as the backbone grows: CODI + SIM-CoT is level with explicit CoT at GPT-2 (42.6 vs. 42.7) yet trails it by 9.2 points at 3B, and KaVa (Kuzina et al., 2026) widens similarly (1.9 to 5.8 points, Table 12). LOTUS instead stays within $\sim$1.5 points of explicit CoT in-domain at each scale, and its gap does not widen. While the smaller-scale results are rather clustered, the decisive comparison is at LLAMA-3.2-3B-INSTRUCT: LOTUS surpasses explicit CoT on the out-of-domain average and brings the in-domain GSM8K gap to within 1.5 points. LOTUS + CODI further tightens it to within 1 point.

The two routings match at 3B, but only direct LOTUS stays robust at smaller scales. At 3B, LOTUS-aux performs comparably to LOTUS and well above the CODI + SIM-CoT baseline, which uses an auxiliary decoder of the same size as the base LM. The gains therefore come from the looped padded backbone with parallel CoT supervision, regardless of the routing. This parity holds only at 3B: at smaller scales LOTUS-aux degrades, whereas direct LM-head supervision (LOTUS) stays robust across scale.

LOTUS reasons $2.5\times$ faster than explicit CoT. The thought phase dominates the difference between methods (Table 2): CoT decodes its chain autoregressively, while LOTUS compresses the same "thinking" into R parallel latent iterations, making its thought $\mathbf{2.5}\times$ faster than CoT and $\mathbf{1.2}\times$ faster than SIM-CoT[3] ($\mathbf{2.1}\times$ faster than CoT in total[4]). CODI is faster than LOTUS in the thought phase because it decodes only a single latent per step, rather than LOTUS's $Kc = 150$ parallel positions, but comes at the cost of accuracy. LOTUS is nonetheless only modestly slower (133 vs. 88 ms) because it consumes all positions in parallel. The inference-time width sweep in Table 7 further separates sequential depth from parallel width: with R fixed, increasing the latent prefix from 6 to 300 positions changes thought latency by only 30 ms.

Speedup grows to $6.9\times$ on natural-language CoT traces. The main GSM8K-Aug setting uses compact math-expression steps, while natural-language CoT requires explicit models to decode much longer reasoning in natural language. In the 3B natural-language CoT stress test (Table 3), LOTUS is on par with explicit CoT in accuracy (68.13 vs. 68.41) while cutting thought-phase latency from 963.6 ms to 140.8 ms with a $6.9\times$ speedup. On this natural-language setting, LOTUS (68.13% GSM8K) far exceeds the latent baselines PCCoT (47.6%), CODI (55.9%), and KaVa (60.0%) reported by Kuzina et al. (2026) (Table 13), staying within variance of explicit CoT (Table 3).

[2] K is chosen to cover the target maximum CoT step count: GSM8K solutions have at most six steps for 99% of examples, so K=6 assigns one block per step and falls back to autoregressive completion otherwise.

[3] We measure latency on a single NVIDIA H100 NVL at batch size 1 with greedy decoding, split into query-prefill, thought, and answer phases. SIM-CoT and CODI both replace each explicit CoT step with a single latent token decoded sequentially. SIM-CoT additionally adds LoRA adapters (rank 128) on top of CODI, which inflates every phase.

[4] LOTUS's prefill and answer phases are just the standard LM operations (Figure 2), so they match explicit CoT in Table 2 (16.7 vs. 15.9 ms and 31.5 vs. 29.5 ms); the looped latent thought is the only phase that differs.

Table 1: Main results across three backbones (GPT-2, LLAMA-3.2-1B-INSTRUCT, LLAMA-3.2-3B-INSTRUCT). Accuracy (%) on in-domain (GSM8K) and out-of-domain (GSM-Hard, MultiArith, SVAMP) benchmarks. All methods are trained on GSM8k-Aug (Deng et al., 2023). "Average" is the mean over the three out-of-domain datasets. Results are reported as mean ± std across 3 seeds. The 3B block is the main scaling comparison: LOTUS nearly closes the in-domain gap to the explicit CoT reference while leading the out-of-domain average.

		In-domain	**Out-of-domain**			
Backbone	Method	GSM8K ↑	GSM-Hard ↑	MultiArith ↑	SVAMP ↑	Average ↑
GPT-2	Explicit CoT	42.7	9.0	85.0	41.6	45.2
	No-CoT	19.1	4.3	41.1	16.4	20.6
	CODI	42.0	9.4	**93.0**	41.7	48.0
	CODI + SIM-CoT	42.6	9.4	92.8	**42.6**	**48.3**
	LOTUS	**44.1** ± 0.7	9.5 ± 0.2	92.4 ± 1.4	41.8 ± 0.9	47.9
	LOTUS + CODI	43.1 ± 1.2	**9.8** ± 0.5	90.0 ± 4.3	41.6 ± 0.4	47.1
	LOTUS-aux	35.5 ± 4.5	8.2 ± 1.2	82.0 ± 2.8	35.5 ± 1.5	41.9
	LOTUS-aux + CODI	38.1 ± 3.7	8.6 ± 0.5	85.7 ± 4.1	36.1 ± 2.6	43.5
LLAMA 3.2-1B	Explicit CoT	58.4	13.9	96.7	65.7	58.8
	No-CoT	28.8	6.3	50.3	26.7	27.8
	CODI	52.7	11.9	95.0	60.6	55.8
	CODI + SIM-CoT	56.1	**12.7**	96.2	**61.5**	56.8
	LOTUS	57.3 ± 0.2	**12.7** ± 0.5	98.3 ± 2.0	60.9 ± 0.6	57.3
	LOTUS + CODI	**58.3** ± 0.8	**12.7** ± 0.2	**98.9** ± 0.6	61.1 ± 0.9	**57.6**
	LOTUS-aux	55.4 ± 0.2	12.6 ± 0.2	97.8 ± 0.0	58.0 ± 0.6	56.1
	LOTUS-aux + CODI	50.6 ± 4.7	11.4 ± 0.8	95.7 ± 1.2	55.0 ± 3.6	54.0
LLAMA 3.2-3B	Explicit CoT	71.5	17.0	98.3	71.0	62.1
	No-CoT	38.3	9.5	88.7	52.9	50.4
	CODI	60.8	14.3	98.7	73.3	62.1
	CODI + SIM-CoT	62.3	14.6	98.8	74.9	62.8
	LOTUS	70.0 ± 0.9	16.0 ± 0.5	**99.9** ± 0.3	**75.7** ± 0.8	**63.9** ± 0.3
	LOTUS + CODI	**70.6** ± 0.2	16.3 ± 0.3	99.6 ± 0.3	74.4 ± 1.4	63.4 ± 0.3
	LOTUS-aux	69.9 ± 0.9	**16.6** ± 0.5	99.8 ± 0.3	74.6 ± 2.2	63.7 ± 0.8
	LOTUS-aux + CODI	70.5 ± 0.7	16.5 ± 0.4	99.4 ± 0.6	73.9 ± 0.1	63.3 ± 0.4

Table 2: Per-phase inference latency (average ms/example) on the GSM8K test set with LLAMA-3.2-3B-INSTRUCT backbone.

Phase	**LOTUS**	Explicit CoT	SIM-CoT	CODI
Query prefill	16.7	15.9	29.5	16.5
Thought	133.0	338.8	162.7	88.2
Answer	31.5	29.5	59.2	30.4
Sum	181.2	384.2	251.4	135.1

4.3 ABLATION STUDIES

We ablate each component of LOTUS to isolate its contribution. Unless noted, all ablations use LLAMA-3.2-3B-INSTRUCT, evaluate on GSM8K test set (Cobbe et al., 2021).

4.3.1 LATENT SUPERVISION DESIGN

LOTUS supervises the latent blocks via a direct readout ($\mathcal{L}_{\text{step}}$) of the *post-loop* latents $\boldsymbol{h}^{(R)}$ through the main LM head f_{head}. We compare it against five alternatives, holding the rest fixed ($K{=}R{=}6$, $c{=}25$, also keeping $\mathcal{L}_{\text{ans}}$): **(i)** no latent supervision (only $\mathcal{L}_{\text{ans}}$); **(ii)** CODI (Shen et al., 2025), distilling a single pre-answer latent onto the teacher CoT model's hidden state; **(iii)** LOTUS (per-iter), reading $\boldsymbol{h}^{(t)}$ through f_{head} at each iteration t rather than the post-loop $\boldsymbol{h}^{(R)}$; **(iv)** LOTUS-aux,

Table 3: Natural-language CoT stress test on GSM8K with the LLAMA-3.2-3B-INSTRUCT. LOTUS is statistically on par with explicit CoT accuracy while reducing thought-phase latency by 6.9×.

Method	GSM8K acc. (%) ↑	Thought ms/example ↓
Explicit CoT	68.41 ± 0.59	963.6 (1.0×)
LOTUS	68.13 ± 0.77	**140.8** (**6.9**×)

Table 4: Latent supervision design comparison.

Latent supervision	Routed via	Test acc. (%) ↑
None (only $\mathcal{L}_{\text{ans}}$)	—	63.3
CODI only	—	64.4
LOTUS (per-iter supervision of $\boldsymbol{h}^{(t)}$)	main LM head	68.2
LOTUS (post-loop supervision of $\boldsymbol{h}^{(R)}$, default)	main LM head	**70.0**
LOTUS-aux (per-iter supervision of $\boldsymbol{h}^{(t)}$, default)	auxiliary decoder	69.9
LOTUS-aux (post-loop supervision of $\boldsymbol{h}^{(R)}$)	auxiliary decoder	68.4

reading $\boldsymbol{h}^{(t)}$ through a separate auxiliary decoder g_ϕ per iteration (default, Appendix C); and **(v)** LOTUS-aux (post-loop), reading $\boldsymbol{h}^{(R)}$ through g_ϕ instead.

Both the looped backbone and latent supervision are beneficial. Without any latent supervision, the looped backbone alone (63.3%) already exceeds CODI + SIM-CoT (62.3%, Table 1), which uses a single latent per step; the looped padded backbone is thus already beneficial on its own. Every supervised variant exceeds both no latent supervision and CODI-only, confirming that the gold CoT supervision contributes to the gain. LOTUS (post-loop direct, 70.0%) matches the best LOTUS-aux schedule (per-iter, 69.9%). The direct (LOTUS) and auxiliary decoder (LOTUS-aux) routings thus perform comparably at their best schedule.

The best supervision schedule differs between LOTUS and LOTUS-aux. LOTUS performs best with post-loop readout (70.0% vs. 68.2% per-iter), which lets the early latent blocks refine fully until the end of the loop. LOTUS-aux instead works best with per-iteration readout (69.9% vs. 68.4% post-loop), which shortens the gradient path—possibly helpful because its extra decoder already lengthens it.

4.3.2 LOOPED ARCHITECTURE DESIGN

Latent token budget per block c. Separate models trained at $c \in \{1, 5, 10, 25, 30\}$ with $K{=}6$ improve from 49.7% at $c{=}1$ to 67.5% at $c{=}5$, then plateau at 70.0% for $c{=}25$ and $c{=}30$ (Table 7). One token per step is too narrow, but moderate widths suffice.

Inference-time c. On the checkpoint trained at $c{=}25$, reducing inference-time c hurts accuracy, while $c{=}30$ and $c{=}50$ slightly improve it to 70.5% (Table 7). Across the 50× width range, thought latency rises only ∼30 ms (111 to 141 ms): all positions are processed in parallel within $R{=}6$ steps, unlike autoregressive CoT.

Looped iterations R. With $K{=}6$ and $c{=}25$ fixed, separately trained models improve from 14.6% at $R{=}2$ to 68.1% at $R{=}5$ and 70.0% at $R{=}6$ (Table 5). Increasing sequential refinement depth is beneficial even at a fixed 150-position latent budget.

Inference-time R. Reusing the $R{=}6$ model with 150 latent positions, accuracy rises from 22.7% at one iteration to 70.0% at six, then dips to 69.3% at seven (Table 6). Fewer iterations limit refinement; extra iterations beyond training bring no further gain.

Latent representation analysis. Due to space constraints, we present the detailed latent representation analysis in Appendix A. A direct readout of LOTUS's post-loop latents recovers gold CoT

Table 5: Training-time looped iterations R (separate models, latent prefix fixed at Kc=150).

R	GSM8K acc. (%) ↑
2	14.6
3	23.2
4	52.6
5	68.1
6	70.0

Table 6: Inference-time looped iterations R (reusing the model trained at R=6).

R	GSM8K acc. (%) ↑
1	22.7
2	40.0
3	55.0
4	63.5
5	68.7
6	70.0
7	69.3

Table 7: Latent token budget c with $K = R = 6$. "Trained" uses separate models; "Inference" reuses the model trained at $c = 25$. Latency is for the inference-time sweep. A dash denotes an unreported trained variant.

c	Kc	Trained acc. (%)	Inference acc. (%)	Thought (ms)
1	6	49.7	51.4	110.9
5	30	67.5	59.4	120.3
10	60	68.4	64.2	127.0
25	150	70.0	70.0	133.0
30	180	70.0	70.5	136.0
50	300	—	70.5	141.2

tokens with 70.9% top-1 accuracy (Table 8). The analysis also examines probability mass on unseen valid reasoning chains and the complementary roles of step and answer supervision (Tables 9 and 10).

5 Conclusion

We introduce LOTUS, showing that latent reasoning can approach the performance of explicit CoT by supervising a looped padded Transformer in parallel against the gold CoT tokens under the simple cross-entropy objective. On LLAMA-3.2-3B-INSTRUCT, LOTUS bridges the in-domain gap to explicit CoT on GSM8K, surpasses CoT on the out-of-domain average, and cuts thought-phase latency by 2.5×. Ablations show the looped backbone, parallel gold CoT supervision, and sufficient block width and loop depth are each necessary. The latent representation analysis further shows the latents are transparent: the gold CoT is recoverable from them by a direct readout, they place graded probability on unseen but valid reasoning chains rather than a single memorized trace, and the step and answer losses contribute complementary structure.

Limitations. We follow prior latent reasoning work (Hao et al., 2025; Shen et al., 2025; Wei et al., 2025) and evaluate on math benchmarks. Whether the recipe transfers to other domains remains an open direction for future work. The block budget K, per-block width c, and loop depth R are fixed hyperparameters that must be set to cover the expected step count, so chains longer than K steps fall back to autoregressive completion of the tail. Making K, c, and R adaptive is a natural and promising direction for future work.

AI USE STATEMENT

We used generative AI tools to polish the manuscript, assist with debugging research code, and help prepare the formatting. These uses did not include generating synthetic datasets, developing the theoretical or conceptual framework, formulating mathematical claims or proofs, proposing hypotheses, designing experiments or research methodology, translating the manuscript, cleaning or reformatting datasets, or interpreting results. The authors manually reviewed the AI-assisted text and code and tested the affected code. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the accuracy, efficiency, and interpretability of latent reasoning on mathematical reasoning benchmarks. These evaluations do not establish reliability in other domains or in high-stakes applications. Recoverable intermediate tokens should not by themselves be interpreted as a guarantee of faithful or correct reasoning. Any deployment would require additional evaluation of errors, robustness, and application-specific risks.

REPRODUCIBILITY STATEMENT

Section 3 specifies the architecture, training objectives, and inference procedure. Section 4.1 describes the datasets, backbones, and comparison methods; Appendix G provides the curriculum, optimization settings, checkpoint selection, hardware, and latency measurement protocol. Additional baseline comparisons and latent-readout examples are provided in Appendices F and I. The main accuracy table reports variation across random seeds.

REFERENCES

Awni Altabaa, Siyu Chen, John Lafferty, and Zhuoran Yang. Unlocking out-of-distribution generalization in transformers via recursive latent space reasoning. *arXiv preprint arXiv:2510.14095*, 2025. URL https://arxiv.org/abs/2510.14095.

Ido Amos, Avi Caciularu, Mor Geva, Amir Globerson, Jonathan Herzig, Lior Shani, and Idan Szpektor. Latent reasoning with supervised thinking states. *arXiv preprint arXiv:2602.08332*, 2026. URL https://arxiv.org/abs/2602.08332.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise loRA. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=WwpYSOkkCt.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=QuqsEIVWIG.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning. *arXiv preprint arXiv:2605.19376*, 2026.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, pp. 5209–5235. PMLR, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Hung-Hsuan Chen. Thinking deeper, not longer: Depth-recurrent transformers for compositional generalization. *arXiv preprint arXiv:2603.21676*, 2026. URL https://arxiv.org/abs/2603.21676.

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*, 2025a. URL https://arxiv.org/abs/2505.16782.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28241–28259, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1369. URL https://aclanthology.org/2025.acl-long.1369/.

Zhikang Chen, Sen Cui, Deheng Ye, Yu Zhang, Yatao Bian, and Tingting Zhu. Think consistently, reason efficiently: Energy-based calibration for implicit chain-of-thought. *arXiv preprint arXiv:2511.07124*, 2025c. URL https://arxiv.org/abs/2511.07124.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024. URL https://arxiv.org/abs/2412.13171.

Yunlong Chu, Minglai Shao, Yuhang Liu, Bing Hao, Yumeng Lin, Jialu Wang, and Ruijie Wang. Spot: Span-level pause-of-thought for efficient and interpretable latent reasoning in large language models. *arXiv preprint arXiv:2603.06222*, 2026. URL https://arxiv.org/abs/2603.06222.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. MoEUT: Mixture-of-experts universal transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZxVrkm7Bjl.

Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Kz6eUL86XP.

Yingqian Cui, Zhenwei Dai, Bing He, Zhan Shi, Hui Liu, Rui Sun, Zhiji Liu, Yue Xing, Jiliang Tang, and Benoit Dumoulin. How do latent reasoning methods perform under weak and strong supervision? *arXiv preprint arXiv:2602.22441*, 2026. URL https://arxiv.org/abs/2602.22441.

DeepSeek-AI. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, Sep 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2019. URL https://arxiv.org/abs/1807.03819.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition. *arXiv preprint arXiv:2510.15522*, 2026. URL https://arxiv.org/abs/2510.15522.

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.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

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

Renyu Fu and Guibo Luo. Selar: Selective latent reasoning in large language models. *arXiv preprint arXiv:2604.08299*, 2026. URL https://arxiv.org/abs/2604.08299.

Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Selective latent iterations to improve reasoning language models. *arXiv preprint arXiv:2511.08577*, 2025. URL https://arxiv.org/abs/2511.08577.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *International Conference on Machine Learning*, pp. 14060–14079. PMLR, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025a. URL https://arxiv.org/abs/2502.05171.

Jonas Geiping, Xinyu Yang, and Guinan Su. Efficient parallel samplers for recurrent-depth models and their connections to diffusion language models. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025b. URL https://openreview.net/forum?id=nA5IRfAfbn.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

Halil Alperen Gozeten, Muhammed Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sTPKDKn5ig.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Yinhan He, Wendy Zheng, Yaochen Zhu, Zaiyi Zheng, Lin Su, Sriram Vasudevan, Qi Guo, Liangjie Hong, and Jundong Li. Semcot: Accelerating chain-of-thought reasoning through semantically-aligned implicit tokens. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1ZuzFUMtx6.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RzYXb5YWBs.

Selim Jerad, Anej Svete, Sophie Hao, Ryan Cotterell, and William Merrill. Context-free recognition with transformers. *arXiv preprint arXiv:2601.01754*, 2026. URL https://arxiv.org/abs/2601.01754.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yian Ma, and Lianhui Qin. Ladir: Latent diffusion enhances LLMs for text reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=z5cPEZ4n6i.

Jonas Knupp, Jan Hendrik Metzen, Jeremias Bohn, Georg Groh, and Kristian Kersting. Depth-recurrent attention mixtures: Giving latent reasoning the attention it deserves. *arXiv preprint arXiv:2601.21582*, 2026. URL https://arxiv.org/abs/2601.21582.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026. URL https://arxiv.org/abs/2604.07822.

Yeskendir Koishekenov, Aldo Lipani, and Nicola Cancedda. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. *arXiv preprint arXiv:2510.07358*, 2025. URL https://arxiv.org/abs/2510.07358.

Anna Kuzina, Maciej Pióro, and Babak Ehteshami Bejnordi. Kava: Latent reasoning via compressed KV-cache distillation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ePrhcLbtGv.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2026a. URL https://arxiv.org/abs/2505.13308.

Jindong Li, Yali Fu, Li Fan, Jiahong Liu, Yao Shu, Chengwei Qin, Menglin Yang, Irwin King, and Rex Ying. Implicit reasoning in large language models: A comprehensive survey. *arXiv preprint arXiv:2509.02350*, 2025. URL https://arxiv.org/abs/2509.02350.

Juncai Li, Ru Li, Yuxiang Zhou, Boxiang Ma, and Jeff Z. Pan. Chain of thought compression: A theoritical analysis. *arXiv preprint arXiv:2601.21576*, 2026b. URL https://arxiv.org/abs/2601.21576.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, pp. 28935–28948. PMLR, 2024a.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=3EWTEy9MTM.

Long Lian, Sida Wang, Felix Juefei-Xu, Tsu-Jui Fu, Xiuyu Li, Adam Yala, Trevor Darrell, Alane Suhr, Yuandong Tian, and Xi Victoria Lin. Threadweaver: Adaptive threading for efficient parallel reasoning in language models. *arXiv preprint arXiv:2512.07843*, 2025. URL https://arxiv.org/abs/2512.07843.

Houjun Liu, Shikhar Murty, Christopher D. Manning, and Róbert Csordás. Thoughtbubbles: an unsupervised method for parallel thinking in latent space. *arXiv preprint arXiv:2510.00219*, 2026. URL https://arxiv.org/abs/2510.00219.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 2025a.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=IaUJl5RCOu.

Kaitlin Maile and João Sacramento. Dynamic parameter reuse augments reasoning via latent chain of thought. In *ICLR Blogposts 2026*, 2026. URL https://iclr-blogposts.github.io/2026/blog/2026/recur-refine-reason/. https://iclr-blogposts.github.io/2026/blog/2026/recur-refine-reason/.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025. URL https://arxiv.org/abs/2511.07384.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

William Merrill and Ashish Sabharwal. Exact expressive power of transformers with padding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=O1abxStFcy.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=5pHfYe10iX.

Ibraheem Muhammad Moosa, Suhas Lohit, Ye Wang, Moitreya Chatterjee, and Wenpeng Yin. Understanding dynamic compute allocation in recurrent transformers. *arXiv preprint arXiv:2602.08864*, 2026. URL https://arxiv.org/abs/2602.08864.

Kei-Sing Ng and Qingchen Wang. Loop neural networks for parameter sharing. *arXiv preprint arXiv:2409.14199*, 2024. URL https://arxiv.org/abs/2409.14199.

Alex Ning, Yen-Ling Kuo, and Gabe Gomes. Learning when to stop: Adaptive latent reasoning via reinforcement learning. *arXiv preprint arXiv:2511.21581*, 2025. URL https://arxiv.org/abs/2511.21581.

Franz Nowak, Anej Svete, Alexandra Butoi, and Ryan Cotterell. On the representational capacity of neural language models with chain-of-thought reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12510–12548, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.676. URL https://aclanthology.org/2024.acl-long.676/.

OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2026. URL https://arxiv.org/abs/2412.16720.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *COLM*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. URL https://arxiv.org/abs/2404.02258.

Michael Rizvi-Martel and Marius Mosbach. The illusion of superposition in latent cot via soft thinking. In *Workshop on Latent & Implicit Thinking – Going Beyond CoT Reasoning*, 2026. URL https://openreview.net/forum?id=FvPx9Nzvnw.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Alok Shah, Khush Gupta, Keshav Ramji, and Pratik Chaudhari. Language modeling with learned meta-tokens. In *ICML 2025 Workshop on Long-Context Foundation Models*, 2025. URL https://openreview.net/forum?id=oaHYnLldHM.

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 Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Shixiang Song, He Li, Zitong Wang, Boyi Zeng, Feichen Song, Yixuan Wang, Zhiqin John Xu, Ziwei He, and Zhouhan Lin. Adaponderlm: Gated pondering language models with token-wise adaptive depth. *arXiv preprint arXiv:2603.01914*, 2026. URL https://arxiv.org/abs/2603.01914.

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. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hYfOPXrbUr.

Ayhan Suleymanzade, Andreas Bergmeister, and Stefanie Jegelka. Limits of continuous chain-of-thought in multi-step and multi-chain reasoning. In *Logical and Symbolic Reasoning in Language Models @ AAAI 2026*, 2026a. URL https://openreview.net/forum?id=UQFTJPqJAc.

Ayhan Suleymanzade, Halil Alperen Gozeten, Ismail Ilkan Ceylan, and Jinwoo Kim. MUX: Continuous reasoning via multiplexed tokens. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026b. URL https://openreview.net/forum?id=Ee78Uqsq0a.

Anej Svete and Ashish Sabharwal. On the reasoning abilities of masked diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BVnIsh4Nz1.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AQsko3PPUe.

Guo Tang, Shixin Jiang, Heng Chang, Nuo Chen, Yuhan Li, Huiming Fan, Jia Li, Ming Liu, and Bing Qin. Looprpt: Reinforcement pre-training for looped language models. *arXiv preprint arXiv:2603.19714*, 2026. URL https://arxiv.org/abs/2603.19714.

Jiecong Wang, Hao Peng, and Chunyang Liu. Latent chain-of-thought as planning: Decoupling reasoning from verbalization. *arXiv preprint arXiv:2601.21358*, 2026. URL https://arxiv.org/abs/2601.21358.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025. URL https://arxiv.org/abs/2509.20317.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with Jacobi iteration. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng

(eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.47. URL https://aclanthology.org/2025.emnlp-main.47/.

Zheyang Xiong, Ziyang Cai, John Cooper, Albert Ge, Vasilis Papageorgiou, Zack Sifakis, Angeliki Giannou, Ziqian Lin, Liu Yang, Saurabh Agarwal, et al. Everything everywhere all at once: LLMs can in-context learn multiple tasks in superposition. *arXiv preprint arXiv:2410.05603*, 2024. URL https://arxiv.org/abs/2410.05603.

Kevin Xu and Issei Sato. A formal comparison between chain-of-thought and latent thought. *ArXiv*, abs/2509.25239, 2025a. URL https://api.semanticscholar.org/CorpusID:281681404.

Kevin Xu and Issei Sato. To CoT or to loop? A formal comparison between chain-of-thought and looped transformers. *arXiv preprint arXiv:2505.19245*, 2025b. URL https://arxiv.org/abs/2505.19245.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025. URL https://arxiv.org/abs/2505.11484.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r1WEQzkCQv.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. *arXiv preprint arXiv:2510.07745*, 2026. URL https://arxiv.org/abs/2510.07745.

Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, Guibin Zhang, Jiale Tao, Jiayi Zhang, Siyuan Ma, Kaituo Feng, Haojie Huang, Youxing Li, Ronghao Chen, Huacan Wang, Chenglin Wu, Zikun Su, Xiaogang Xu, Kelu Yao, Kun Wang, Chen Gao, Yue Liao, Ruqi Huang, Tao Jin, Cheng Tan, Jiangning Zhang, Wenqi Ren, Yanwei Fu, Yong Liu, Yu Wang, Xiangyu Yue, Yu-Gang Jiang, and Shuicheng Yan. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026. URL https://arxiv.org/abs/2604.02029.

Boyi Zeng, He Li, Shixiang Song, Yixuan Wang, Zitong Wang, Ziwei He, Xinbing Wang, and Zhouhan Lin. Ponderlm-2: Pretraining llm with latent thoughts in continuous space. *arXiv preprint arXiv:2509.23184*, 2026a. URL https://arxiv.org/abs/2509.23184.

Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu li, and Zhouhan Lin. PonderLM: Pretraining language models to ponder in continuous space. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=UrM4MNRYZm.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *arXiv preprint arXiv:2512.21711*, 2025. URL https://arxiv.org/abs/2512.21711.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *ICML 2025 Workshop on Methods and Opportunities at Small Scale*, 2025a. URL https://openreview.net/forum?id=1cD9iO5Isv.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025b. URL https://arxiv.org/abs/2507.06203.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025c. URL https://arxiv.org/abs/2510.25741.

Jiaxuan Zou, Yaozhong Xiong, and Yong Liu. Capabilities and fundamental limits of latent chain-of-thought. *arXiv preprint arXiv:2602.01148*, 2026. URL https://arxiv.org/abs/2602.01148.

APPENDIX

Appendix A presents the latent representation analysis deferred from the main paper due to space constraints. Appendices B and C provide the supporting loss-factorization analysis and auxiliary-decoder formulation. Appendix D discusses related works including looped transformers, latent reasoning, and speculative decoding methods. Appendix E summarizes the notation used in the paper. Appendix F compares LOTUS and LOTUS-aux with the closest latent reasoning baselines. Appendix G includes the experimental and implementation details. Appendix H explains the relation between PCL and autoregressive NLL lower bounds. Finally, Appendix I provides the latent representation analysis details, including greedy readout examples and the multi-path setup and examples.

A LATENT REPRESENTATION ANALYSIS

We analyze the learned latent representations along three axes. Together, these analyses show that LOTUS's accuracy gains are accompanied by structured latent computation: the post-loop latents carry readable CoT token signal, assign nontrivial mass to unseen valid alternatives, and rely on both gold CoT supervision and answer supervision to form coherent reasoning states. Appendix A.1 measures the likelihood assigned to the gold CoT under different readouts, Appendix A.2 tests whether the latents place mass on unseen but valid reasoning chains, and Appendix A.3 ablates the roles of gold CoT and answer supervision in shaping the representation. Extended block-to-step decode examples are deferred to Appendix I.1.

A.1 LIKELIHOOD OF THE GOLD CHAIN UNDER DIFFERENT READOUT OPTIONS

Recall that $\boldsymbol{h}^{(R)}$ denotes the post-loop latents and $\widetilde{\boldsymbol{z}}^{(t)}$ the aux decoder's output latents for $t \in \{1, \ldots, R\}$. We evaluate four readouts in Table 8: **(i)** LOTUS, reading $\boldsymbol{h}^{(R)}$ through the base LM head; **(ii)** LOTUS-aux, reading $\boldsymbol{h}^{(R)}$ through g_{head}; **(iii)** LOTUS-aux, reading $\widetilde{\boldsymbol{z}}^{(t)}$ through g_{head} under *teacher forcing* (TF) that matches training (each block's c latents paired with that block's gold step tokens); and **(iv)** LOTUS-aux, reading $\widetilde{\boldsymbol{z}}^{(t)}$ through g_{head} but under *free-running* (FR),[5] feeding the aux decoder's own generated token in place of the gold tokens.[6]

Metrics. We report the negative log-likelihood (NLL) of the gold CoT token sequence T at the latent positions.[7] We also report average $P(\text{gold} \in \text{top-1})$ and $P(\text{gold} \in \text{top-5})$ against the full 128K-vocabulary softmax. All metrics are averaged over the 1,319 GSM8K test questions.

LOTUS's post-loop latent $\boldsymbol{h}^{(R)}$ contains the gold CoT via a direct readout. Reading LOTUS's post-loop latent $\boldsymbol{h}^{(R)}$ through f_{head} assigns the gold CoT a low 3.07 NLL and 70.9% top-1 accuracy (Table 8), edging out LOTUS-aux (FR) at 64.5%. The teacher-forced LOTUS-aux reader does better still (1.56 NLL, 84.3% top-1), but with the caveat that it uses the gold prefix as input, so we read it as an upper bound rather than a direct comparison. Still, the results confirm that the latents carry enough information to reconstruct the gold CoT chain.

LOTUS-aux has CoT signal in both $\boldsymbol{h}^{(R)}$ and $\widetilde{\boldsymbol{z}}^{(t)}$. Surprisingly, even though $\boldsymbol{h}^{(R)}$ is never directly supervised to predict the gold CoT through g_{head} in LOTUS-aux, we can still read $\boldsymbol{h}^{(R)}$ through g_{head} with a 25.8% top-5 probability. Greedy decoding also surfaces some meaningful gold tokens, indicating that the CoT content is carried by the latents themselves. In fact, despite being

[5] LOTUS-aux (FR) gives a fairer reference to LOTUS than LOTUS-aux (TF): at inference the gold CoT is unavailable, so LOTUS-aux must feed its decoder its own generated tokens, exactly the FR setting. The TF row instead feeds the gold prefix the model would not have at test time.

[6] Here we report greedy decoding results. Replacing them with temperature-T sampling ($T \in \{0.5, 0.7, 1.0, 1.5\}$) moves FR NLL by $\leqslant 0.05$ and top-1 by $\leqslant 1$ pp.

[7] Note that the $\boldsymbol{h}^{(R)}$ readout scores the gold CoT under the PCL (Appendix B) factorization and $\widetilde{\boldsymbol{z}}^{(t)}$ under an autoregressive one, so NLL is only directly comparable under the same factorization and only for reference across factorizations. Mathematically, PCL NLL has a higher lower bound than the autoregressive NLL (details in Appendix H).

Table 8: Likelihood of the gold CoT from different readout configurations.

Reader	NLL	$P(\text{gold} \in \text{top-1})$	$P(\text{gold} \in \text{top-5})$
Without gold prefix:			
LOTUS: $\boldsymbol{h}^{(R)}$ through f_{head}	3.07	70.9%	85.8%
LOTUS-aux: $\boldsymbol{h}^{(R)}$ through g_{head}	9.02	12.4%	25.8%
LOTUS-aux: $\tilde{\boldsymbol{z}}^{(t)}$ through g_{head} (FR)	6.39	64.5%	76.7%
With gold prefix:			
LOTUS-aux: $\tilde{\boldsymbol{z}}^{(t)}$ through g_{head} (TF)	1.56	84.3%	93.9%

supervised through different heads, LOTUS and LOTUS-aux have a *similar* effect of making the post-loop latents $\boldsymbol{h}^{(R)}$ put mass on the CoT steps, with the caveat that LOTUS-aux readouts are less formatted (which is expected since it is not supervised directly). We provide side-by-side $\boldsymbol{h}^{(R)}$ readout examples for both models in Appendix I.

A.2 Mass on Unseen Valid Chains

Beyond the gold chain in the test set, we test whether LOTUS and LOTUS-aux place mass on unseen valid reasoning chains rather than only memorizing the seen trace in the training set. We use the same four readout configurations as in Appendix A.1.

Setup. Given each question, we compare its ground-truth reasoning chain against an unseen-but-valid alternative. Let G and U denote the sets of intermediate numbers appearing uniquely in the ground-truth and the unseen reasoning chain, respectively. A random control set serves as a baseline. We report each set's average per-token negative log-likelihood (NLL) at the position where it surfaces most strongly (lower means more confident). We also report $P(U \in \text{top-}k)$, the fraction of U numbers that surface within the top-k readout at any latent position. See Appendix I.2 for the full construction and metric details.

Findings. Across all four readouts the ordering $G \ll U \ll$ Random holds (Table 9, lower NLL $=$ surfaces more): each reader assigns much lower NLL to a ground-truth-only number than to an unseen-but-valid one, which in turn sits far below the random control. Crucially, the U signal is concentrated rather than diffuse: $P(U \in \text{top-}k)$ columns confirm the unseen-valid intermediates genuinely surface in the top-k (e.g., for LOTUS through the base LM head, 15.3% are top-1 and 64.0% fall within top-5, despite never being trained on). Appendix I.3 and Figure 4 give a path-level example and visual grid: two valid chains reach the same answer (108) through disjoint intermediates, and the unseen-path numbers (24, 18) surface in the post-loop readout despite never appearing in the trained chain, question, or answer.

Notably, this multi-path signal emerges without explicit superposition training: our latent supervision targets the provided gold CoT, yet the latent readouts also assign mass to intermediates from unseen valid alternatives. This echoes Xiong et al. (2024), who show that task superposition can emerge even when models are trained to in-context learn only one task at a time.

A.3 Disentangling the Two Losses in LOTUS

We now ask what each of LOTUS's two supervision losses contributes to the latent representation observed above. The theory in Appendix B ascribes complementary roles to $\mathcal{L}_{\text{step}}$ and $\mathcal{L}_{\text{ans}}$. We compare three models that differ in supervision: LOTUS (both losses), $\mathcal{L}_{\text{step}}$ only, and $\mathcal{L}_{\text{ans}}$ only.

Gold-chain NLL. Reading the $\boldsymbol{h}^{(R)}$ through the base LM head f_{head}, Table 10 reports NLL, $P(\text{gold} \in \text{top-1})$, and $P(\text{gold} \in \text{top-5})$ on the GSM8K test set, using the same protocol as Table 8.

$\mathcal{L}_{\text{step}}$ and $\mathcal{L}_{\text{ans}}$ play complementary roles. LOTUS beats both ablations (Table 10). $\mathcal{L}_{\text{step}}$-only, lacking the cross-block coupling and the answer grounding that $\mathcal{L}_{\text{ans}}$ supplies, is worst on every column. $\mathcal{L}_{\text{ans}}$-only, without directly tying $\boldsymbol{h}^{(R)}$ to the gold CoT, lands nearer the gold tokens (NLL

Table 9: Evaluation on both seen and unseen valid CoT chains. The final two columns report $P(U \in \text{top-}k)$ for $k = 1$ and $k = 5$.

Reader	G NLL	U NLL	Random NLL	Top-1	Top-5
Without gold prefix:					
LOTUS: $\boldsymbol{h}^{(R)}$ through f_{head}	0.07	4.28	8.16	15.3%	64.0%
LOTUS-aux: $\boldsymbol{h}^{(R)}$ through g_{head}	3.30	6.04	9.42	14.5%	49.3%
LOTUS-aux: $\widetilde{\boldsymbol{z}}^{(t)}$ through g_{head} (FR)	0.80	12.62	20.39	5.6%	39.2%
With gold prefix:					
LOTUS-aux: $\widetilde{\boldsymbol{z}}^{(t)}$ through g_{head} (TF)	0.18	13.17	20.83	1.9%	37.4%

Table 10: Disentangling on the two loss components; NLL and per-position gold CoT accuracy on the GSM8K test.

Variant	NLL ↓	$P(\text{gold} \in \text{top-1})$ ↑	$P(\text{gold} \in \text{top-5})$ ↑
LOTUS	3.07	70.9%	85.8%
$\mathcal{L}_{\text{step}}$ only	9.29	9.1%	17.1%
$\mathcal{L}_{\text{ans}}$ only	5.97	9.4%	26.5%

5.97, top-5 26.5%) yet has a top-1 rate similar to $\mathcal{L}_{\text{step}}$-only and stays far short of LOTUS (NLL 3.07, top-5 85.8%). The answer loss alone settles $\boldsymbol{h}^{(R)}$ into an answer-consistent neighborhood without aligning to the gold chain, matching the complementary roles posited in Appendix B.

B Roles of $\mathcal{L}_{\text{step}}$ and $\mathcal{L}_{\text{ans}}$

A natural worry about the LOTUS objective (Equation (6)) is that it supervises each latent position *independently*: Equation (3) scores every gold CoT token in parallel, with no autoregressive factorization of the chain. Why should independent per-position targets train a model that produces a globally coherent answer? We give a lens that resolves this, distinguishing the role of the two losses and explaining why both are needed.

Parallel chain likelihood. The step loss $\mathcal{L}_{\text{step}}$ supervises a *parallel chain likelihood* (PCL) over the readout positions. Maximizing the per-position probability $p_{\boldsymbol{\theta}}(T_{i,j} \mid Q)$ of each gold token at its latent position induces the chain likelihood

$$p_{\boldsymbol{\theta}}^{\text{PCL}}(T \mid Q) = \prod_{i=1}^{K} \prod_{j=1}^{c} p_{\boldsymbol{\theta}}(T_{i,j} \mid Q), \tag{7}$$

rather than the autoregressive conditionals $p_{\boldsymbol{\theta}}(T_{i,j} \mid Q, T_{<i}, T_{i,<j})$. The factorization treats the readout as conditionally independent across positions, but the latent states themselves are not independent: the looped Transformer computes them *jointly* over the padded workspace, so the dependence the factorization drops is carried by the shared latent computation rather than by the loss.

Complementary roles. This view makes the division of labor between the two losses precise. First, the support inclusion

$$\text{supp}(q(T \mid Q)) \subseteq \prod_{i=1}^{K} \prod_{j=1}^{c} \text{supp}(q(T_{i,j} \mid Q)) \tag{8}$$

shows that every gold chain lies inside the Cartesian product of per-position gold token supports.[8] So $\mathcal{L}_{\text{step}}$ plays a *support-coverage* role: it makes each position place mass on the right gold tokens,

[8] Minimizing $\mathcal{L}_{\text{step}}$, the cross-entropy to these marginals, pulls the model's PCL toward that product, i.e. minimizes $\text{KL}\big(\prod_{i,j} q(T_{i,j} \mid Q) \,\|\, p_{\boldsymbol{\theta}}^{\text{PCL}}(T \mid Q)\big)$.

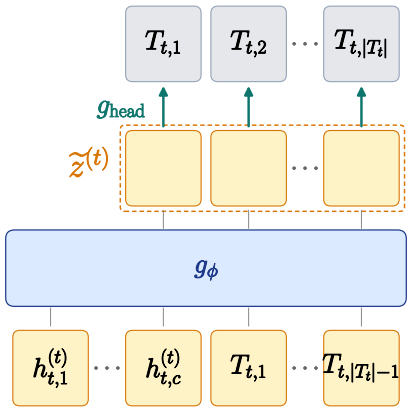


Figure 3: LOTUS-aux supervision at loop iteration t for block t. The auxiliary decoder g_ϕ only replaces the base LM head supervision in Figure 2(a).

without needing to reproduce the joint distribution—which LOTUS never samples, since the answer is decoded directly from the jointly computed latent states. Second, $\mathcal{L}_{\text{ans}}$ supplies the global *selection* pressure that PCL alone lacks: because the answer is trained while conditioning on the entire latent configuration, gradients favor jointly computed hidden states that can actually support the correct answer. The two losses are thus complementary by construction—coverage from $\mathcal{L}_{\text{step}}$, joint selection from $\mathcal{L}_{\text{ans}}$—which is exactly the behavior we verify empirically in Appendix A.3, where dropping either loss degrades the gold chain likelihood.

To further probe how LOTUS compares to modeling the autoregressive chain likelihood, we introduce an autoregressive decoder variant in Appendix C and compare the results in Section 4.2.

C LOTUS-AUX: AN AUXILIARY DECODER VARIANT

An alternative route for CoT supervision explored in existing work is autoregressive chain likelihood: an auxiliary autoregressive decoder conditioned on the latent blocks scores the CoT tokens under teacher forcing. This creates a trade-off: in exchange for modeling the chain autoregressively, CoT supervision no longer flows directly through the answer-generating base LM head, and teacher forcing introduces an extra train/inference mismatch.

We instantiate this routing on our looped padded backbone as LOTUS-aux. LOTUS-aux reuses the same looped forward and final forward as LOTUS (Figure 2). The only difference is the supervision in the looped forward, which is routed through an auxiliary decoder rather than read directly through the base LM head. The auxiliary decoder mirrors SIM-CoT (Wei et al., 2025), but instead of generating latents autoregressively and supervising a single latent per CoT step, LOTUS-aux produces all K latent blocks in parallel through the looped backbone and supervises a full c-token block per step. We describe this routing below.

Latent supervision via the auxiliary decoder. As shown in Figure 3, the auxiliary decoder g_ϕ is an extra decoder model that, at each loop iteration t, reads latent block $\boldsymbol{h}^{(t)}{}_t$ (we select block $i = t$ at iteration t, the c post-loop latents of the t-th block) as a c-token prefix and predicts the gold CoT step $T_t = (T_{t,1}, \ldots, T_{t,|T_t|})$ under teacher forcing, where $|T_t|$ is the step's token length. Let $\widetilde{\boldsymbol{z}}^{(t)} = (\widetilde{\boldsymbol{z}}^{(t)}_0, \ldots, \widetilde{\boldsymbol{z}}^{(t)}_{|T_t|-1})$ denote the output hidden states at each input CoT position $m \in \{0, 1, \ldots, |T_t| - 1\}$ (m=0 being the last latent position $\boldsymbol{h}^{(t)}_{t,c}$, which predicts $T_{t,1}$, and m>0 the gold

token $T_{t,m}$). Mirroring the final-forward pass of Equation (4), we generate $\widetilde{\boldsymbol{z}}^{(t)}$ with

$$\widetilde{\boldsymbol{z}}^{(t)} = g_\phi\Big([\,\boldsymbol{h}_{t,c}^{(t)},\ T_{t,1},\ldots,T_{t,|T_t|-1}\,] \mid \boldsymbol{h}_{t,1}^{(t)},\ldots,\boldsymbol{h}_{t,c-1}^{(t)}\Big), \tag{9}$$

where the first $c-1$ latent positions $\boldsymbol{h}_{t,1}^{(t)},\ldots,\boldsymbol{h}_{t,c-1}^{(t)}$ are the conditioning prefix. The teacher-forced next-token cross-entropy on the gold CoT gives:

$$\mathcal{L}_{\text{step}}^{\text{aux}} = \frac{1}{N_{\text{step}}}\sum_{t=1}^{K}\sum_{m=0}^{|T_t|-1}\mathrm{CE}\Big(g_{\text{head}}(\widetilde{\boldsymbol{z}}_m^{(t)}),\ T_{t,m+1}\Big), \tag{10}$$

where $N_{\text{step}} = \sum_{t=1}^{K}|T_t|$ is the total number of supervised CoT tokens. The supervision itself remains parallel: under teacher forcing the gold prefix is fed at every position, so all CoT positions are scored in a single forward pass with no sequential generation. In other words, the autoregressive factorization affects only the loss, not the computation.

Per-iteration supervision and the full objective. Equation 10 reads block $i = t$ at iteration t (per-iteration, rather than at the final iteration R). We validate this choice in Section 4.3.1, where per-iteration readouts outperform reading all blocks at the final iteration R for the auxiliary decoder routing. The answer loss $\mathcal{L}_{\text{ans}}$ from Section 3.3 is unchanged, so the full LOTUS-aux objective is $\mathcal{L}^{\text{aux}} = \mathcal{L}_{\text{ans}} + \lambda_{\text{step}}\mathcal{L}_{\text{step}}^{\text{aux}}$. In our experiments g_ϕ is initialized with a copy of the base LM and trained together with the base LM. Training and architectural details are in Appendix G.

Inference. The auxiliary decoder g_ϕ is used only at training time. At inference, LOTUS-aux generates the final answer autoregressively from the post-loop latents exactly as LOTUS does (Figure 2(b)), so the inference efficiency gain of LOTUS also applies to LOTUS-aux.

D Related Work

We refer the reader to Chen et al. (2025a); Yu et al. (2026); Zhu et al. (2025b); Li et al. (2025); Feng et al. (2025), and Maile & Sacramento (2026) for broad surveys of the latent reasoning and looped transformers landscapes. Below, we discuss the work most directly adjacent to ours.

Looped and recurrent-depth transformers. Building on Universal Transformers (Dehghani et al., 2019), a line of work has established the theoretical foundations of looped architectures: Giannou et al. (2023) show Turing completeness via program simulation, placing looped transformers alongside CoT in expressivity (Merrill & Sabharwal, 2024; Li et al., 2024b; Nowak et al., 2024). Saunshi et al. (2025); Merrill & Sabharwal (2025b;a); Jerad et al. (2026) study the efficient reasoning abilities of looping, establishing their ability to solve problems with fewer model evaluations than CoT. Saunshi et al. (2025) show that a shallow Transformer looped multiple times nearly matches a non-looped Transformer with the equivalent total depth on arithmetic and reasoning tasks, with looping implicitly simulating hidden reasoning steps. Xu & Sato (2025b;a) compare looped Transformers to CoT, and Svete & Sabharwal (2026) draw connections to masked diffusion models. The latter connection also inspires approaches to improve looped transformers with time modulation (Jeddi et al., 2026; Chen, 2026) and efficient sampling (Geiping et al., 2025b). An active line of work scales looped models to large-scale pretraining: Geiping et al. (2025a) train a 3.5B recurrent-depth Transformer whose loop depth can be increased at test time, Zhu et al. (2025c) pretrain LoopLM with token-level recurrence and learned depth allocation, and Zeng et al. (2026b;a) pretrain models that loop by superpositioning symbols in continuous space based on the logits produced after each iteration. LoopLM shows that looped computation can scale in pretraining, but its per-token loop at inference time does not by itself create the parallel padded latent workspace that LOTUS uses. Complementary work addresses different ways of implementing *adaptive compute* with looped architectures: Depth can be modulated per-symbol via gating (Ng & Wang, 2024; Song et al., 2026; Moosa et al., 2026), mixture-of-depths routing (Raposo et al., 2024), or RL-trained stopping criteria (Ning et al., 2025). Depth-recurrent attention mixtures augment recurrence with attention mechanisms that can attend to latent representations of past iterations (Fu et al., 2025; Knupp et al., 2026).

Parameter efficiency and flexibility are addressed by per-iteration LoRA adapters (Bae et al., 2025a), symbol-level recursion routing (Bae et al., 2025b), inner thinking mechanisms (Chen et al., 2025b), and Mixture-of-Experts compatibility (Csordás et al., 2024). Further work explores retrofitting recurrence into pretrained models (McLeish et al., 2025; Koishekenov et al., 2025), compositional and length generalization (Fan et al., 2025; Kohli et al., 2026), reinforcement pretraining (Tang et al., 2026), and whether pretrained models use their depth efficiently (Csordás et al., 2025).

Diffusion language models. Diffusion language models are another route to parallel, non-autoregressive reasoning. Discrete-token variants face a *parallel-decoding trap*, where tokens denoised in parallel are conditionally independent, so recovering quality forces a partial return to sequential decoding (Liu et al., 2025a). Notably, Svete & Sabharwal (2026) show masked diffusion models are equivalent to padded looped Transformers, motivating supervising such a backbone directly, as we do. Continuous diffusion variants such as LaDiR (Kang et al., 2026) instead refine latent thoughts with a VAE and a separate diffusion model over many denoising steps for diversity and adaptive compute, while showing limited efficiency gain when reaching CoT-level performance.

Pause and padding symbols. A parallel strand of work studies the affordances of discrete padding symbols. Goyal et al. (2024) and Pfau et al. (2024) show that they improve performance on natural language and formal tasks; Wang et al. (2024) use discrete symbols in the continuous representation space as planning signals; and Chu et al. (2026) introduce span-level pause symbols for efficient and interpretable latent reasoning. These discrete-symbol approaches motivate our use of continuous latent vectors as a richer computation medium, while our looped architecture lets each iteration refine the same latent rather than extending the sequence.

Compressing and replacing discrete CoT. Latent CoT methods replace or compress explicit reasoning symbols with continuous representations. Cheng & Durme (2024) supervise hidden-state representations of salient reasoning steps, Tan et al. (2025) dynamically compress CoT chains into latent thoughts, Wang et al. (2026) autoregressively encode CoT steps into hidden states decoded back to ground-truth symbols (analogously to our approach), and Amos et al. (2026) interleave supervised latent states with generated text. MUX (Suleymanzade et al., 2026b) studies continuous reasoning with multiplexed vocabulary-space targets. It uses KL regression to multiplexed targets. Each target is a position-weighted superposition over aligned reasoning subwords, so several subwords are mixed into one sequential latent. In contrast, LOTUS keeps step and token positions explicit in a parallel padded workspace and applies per-position cross-entropy on gold CoT traces. Latent CoT also enables parallel exploration of reasoning paths (Gozeten et al., 2026; Liu et al., 2026), and test-time compute can be scaled in latent space via policy optimization (Ye et al., 2026), instance-level policy gradient (Li et al., 2026a), sampling conditioned on perturbed prompts (Xu et al., 2025), or parallel search (Lian et al., 2025; You et al., 2026). Latent reasoning can be integrated with generation by mixing latent and text symbols (Fu & Luo, 2026; Deng et al., 2026), updating the KV cache with an additional model (Liu et al., 2025b), learning meta-symbols that guide generation (Shah et al., 2025), and semantically aligning padding symbols to spans of CoT symbols (He et al., 2025). Further work studies latent reasoning with variational autoencoder features (Su et al., 2025; Kang et al., 2026) and energy-based guidance (Chen et al., 2025c). GRAM (Baek et al., 2026) guides thinking with recursive latent-variable modeling through shared transitions. It uses stochastic high-level updates and learns a target-conditioned posterior and prior through amortized variational inference. GRAM samples distinct stochastic trajectories, whereas LOTUS represents them within a jointly refined continuous workspace. Interpretability studies probe whether latents actually contribute to reasoning (Zhang et al., 2025; Rizvi-Martel & Mosbach, 2026), and empirical comparisons examine the effects of different supervision regimes (Cui et al., 2026; Suleymanzade et al., 2026a).

Speculative decoding. Speculative decoding uses draft models to propose token continuations (Leviathan et al., 2023; Chen et al., 2023). Related approaches include Lookahead (Fu et al., 2024) with Jacobi-style updates, Medusa (Cai et al., 2024) with auxiliary prediction heads, and EAGLE (Li et al., 2024a) with future hidden features. These methods parallelize proposals within an explicit token trace, whereas LOTUS parallelizes latent reasoning and decodes only the answer suffix.

Table 11: Notation used in this paper.

Symbol	Meaning	First used
Inputs and targets		
Q	Question	§2
A	Answer suffix (ends with $\langle$EoS$\rangle$)	§3.1
T_i	Tokenized gold CoT step i ($T_{i,j}$ its j-th token)	§2
T	Full gold CoT token sequence $(T_1, \ldots, T_K)$	§3.3
Counts and budgets (fixed vs. example-dependent)		
S	Number of CoT steps in an example (*example-dependent*)	§2
N	Total number of CoT tokens in an example	§2
K	Latent block budget (*fixed*; chosen so $K \geqslant S$)	§3.1
c	Latent tokens per block (block width)	§3.1
R	Number of loop iterations (depth)	§3.2
d	Hidden / embedding dimension of the base LM	§3.2
Indices		
i	Latent block index, $i \in \{1, \ldots, K\}$	§3.1
t	Loop pass index, $t \in \{1, \ldots, R\}$	§3.2
j	Token index within a CoT step / latent block	§2
Model and special tokens		
$f_{\boldsymbol{\theta}}$	Base causal LM (parameters $\boldsymbol{\theta}$)	§3.2
f_{head}	Base LM head (logits from a hidden state)	§3.3
g_ϕ	Auxiliary decoder (LOTUS-aux, training only); head g_{head}	§C
g_{head}	Auxiliary decoder LM head	§C
$\langle$`lat`$\rangle$	Learnable latent token (shared across positions)	§3.1
$\langle$`BoT`$\rangle$, $\langle$`EoT`$\rangle$	Begin-/end-of-thought delimiters	§3.1
$\mathcal{C}_{\text{pre}}$	Prefix KV cache (computed once, reused)	§3.2
$\boldsymbol{E}$	Learnable latent embeddings fed to the loop	§3.2
Hidden states (distinct symbols by source)		
$\boldsymbol{h}^{(t)}$	Post-loop latent states after pass t (entries $\boldsymbol{h}^{(t)}_{i,j}$)	§3.2
$\boldsymbol{z}$	Final-forward hidden states for the answer loss	§3.3
$\widetilde{\boldsymbol{z}}^{(t)}$	Auxiliary-decoder hidden states at iteration t	§C
Losses and objective		
$\mathcal{L}_{\text{step}}$	Parallel step supervision loss (Equation (3))	§3.3
$\mathcal{L}_{\text{ans}}$	Answer-suffix next-token loss	§3.3
$\mathcal{L}$	Full objective, $\mathcal{L}_{\text{ans}} + \lambda_{\text{step}}\,\mathcal{L}_{\text{step}}$	§3.3
λ_{step}	Weight on the step loss	§3.3
N_{step}	Number of supervised (non-padding) CoT tokens	§3.3
$\text{CE}(\cdot, \cdot)$	Cross-entropy	§3.3
$\mathcal{L}^{\text{aux}}_{\text{step}}$	Auxiliary-decoder step loss (Equation (10))	§C
$\mathcal{L}^{\text{aux}}$	LOTUS-aux objective, $\mathcal{L}_{\text{ans}} + \lambda_{\text{step}}\,\mathcal{L}^{\text{aux}}_{\text{step}}$	§C
Analysis (Appendix B)		
Q	Question random variable	§B
q	True data distribution over chains	§B
$p^{\text{PCL}}_{\boldsymbol{\theta}}$	Parallel chain likelihood induced by $\mathcal{L}_{\text{step}}$	§B

E Notation

Table 11 summarizes the notation used throughout the paper.

F Closely Related Latent Reasoning Methods

We position LOTUS and its auxiliary-decoder variant LOTUS-aux against the most closely related latent reasoning methods.

Coconut (Hao et al., 2025) introduced a curriculum-based approach that progressively replaces explicit CoT steps with latent tokens conditioned autoregressively on the previous latent. Both

Table 12: Comparison with latent reasoning methods in other step regimes on GSM8K (trained on GSM8k-Aug).

Method	GPT-2	LLAMA-3.2-1B	LLAMA-3.2-3B
Explicit CoT	42.7	58.4	71.5
Coconut (Hao et al., 2025)	36.6	33.2	–
Coconut + SIM-CoT (Wei et al., 2025)	44.8	42.2	–
PCCoT (Wu et al., 2025)	**49.5**	54.2	54.7
KaVa (Kuzina et al., 2026)	–	56.5	65.7
LOTUS	44.1	**57.3**	**70.0**

LOTUS and LOTUS-aux depart from Coconut in two ways: *(i)* they use a *padded* latent prefix processed by a *looped* Transformer, so all K steps are decoded in parallel, not autoregressively, giving the $\mathcal{O}(R)$-iteration thought-phase cost, and *(ii)* they add step-aligned supervision on the latent blocks, where Coconut supervises only the answer. Our ablation without latent supervision (Table 4) is the Coconut-style regime within our looped architecture, and it underperforms both variants.

CODI (Shen et al., 2025) and SIM-CoT (Wei et al., 2025) retain Coconut's autoregressive latent decoding but add an auxiliary supervision target: CODI aligns the hidden state of a single designated token with the corresponding teacher hidden state from a CoT model, and SIM-CoT trains an auxiliary autoregressive decoder that aligns each latent token with the corresponding CoT token. Our LOTUS removes the auxiliary path entirely, supervising the model's own post-loop hidden states through the answer LM head at block granularity. LOTUS-aux keeps an auxiliary autoregressive decoder, but produces all blocks in parallel through the looped backbone and supervises a full block per step rather than one latent per step.

Parallel Continuous CoT (PCCoT) (Wu et al., 2025) and KaVa (Kuzina et al., 2026) form a single Jacobi-iteration family: PCCoT introduces the backbone and KaVa builds on it. They both refine a fixed budget of 24 latent tokens over 3 Jacobi iterations and differ only in supervision. Like LOTUS, they process latent symbols in parallel and refine them over passes, but differ from our methods in three ways: *(i) Architecture*: PCCoT and KaVa use right-shifted Jacobi iterations to simulate sequential continuous CoT in parallel (Hao et al., 2025), so their latents are continuous thought vectors with no fixed alignment to individual gold CoT steps. LOTUS instead pins block i to fixed positions across all loops, so it always maps to gold CoT step i and is naturally supervised against that step's tokens; *(ii) Supervision*: PCCoT uses CODI (Shen et al., 2025), and KaVa supervises indirectly by distilling a compressed teacher KV-cache, whereas both our variants supervise each latent block against its exact gold CoT step; and *(iii) Depth*: PCCoT reports accuracy peaking at about 3 Jacobi iterations and degrading beyond that, which is why KaVa inherits the 3-iteration setting, whereas LOTUS improves monotonically with loop depth up to 6 (Table 5).

Comparison with latent methods at other compute budgets. The main results (Table 1, Section 4.2) compare LOTUS against latent methods at the *same* sequential step budget (CODI and CODI + SIM-CoT, both R=6). Here we provide a more comprehensive accuracy comparison in Table 12 against latent methods whose sequential compute budget instead differs from LOTUS's R=6 sequential steps. Coconut and Coconut + SIM-CoT decode latents autoregressively under the curriculum of Hao et al. (2025) (10 autoregressive latent tokens), while PCCoT and KaVa decode in parallel over a shared Jacobi-iteration backbone (24 latent tokens in total, refined over 3 iterations). We take accuracies from Wei et al. (2025), Kuzina et al. (2026), and Wu et al. (2025) for each method. LOTUS attains higher in-domain GSM8K accuracy than KaVa at both shared scales (57.3 vs. 56.5 at 1B, 70.0 vs. 65.7 at 3B), staying within 1.5 points of explicit CoT while PCCoT and KaVa fall further behind as the backbone grows. The missing entries follow what each source reports: Wei et al. (2025) evaluate Coconut only up to LLAMA-3.2-1B, attributing Coconut's failure at larger scale to a latent-instability issue in which the latent representations become homogeneous and lose semantic diversity, causing training to collapse as the latent budget grows. Kuzina et al. (2026) do not run KaVa on GPT-2.

Table 13: Comparison with latent reasoning methods on GSM8K, trained on GSM8k-Aug (the natural language version in Wu et al. (2025)).

Method	GSM8K acc. (%) ↑	GSM-Hard acc. (%) ↑	SVAMP acc. (%) ↑
Explicit CoT	68.41 ± 0.59	18.27 ± 0.85	71.93 ± 1.62
PCCoT	47.6	11.0	65.2
CODI	55.9	13.6	70.1
KaVa	60.0	14.8	66.1
LOTUS	$\mathbf{68.13 \pm 0.77}$	$\mathbf{16.27 \pm 0.19}$	$\mathbf{73.40 \pm 0.35}$

Natural-language CoT stress-test. For the natural-language CoT stress test in Section 4.2, we use the LLAMA-3.2-3B-INSTRUCT backbone. The explicit CoT run reaches 68.41% GSM8K accuracy with 963.6 ms thought latency. The looped run reaches 68.13% accuracy with 140.8 ms thought latency. The latent baselines trail well behind on GSM8K (PCCoT 47.6, CODI 55.9, KaVa 60.0). Out of domain, LOTUS leads all methods on SVAMP (73.40, above explicit CoT's 71.93) and stays close to explicit CoT on GSM-Hard (16.27 vs. 18.27), ahead of every latent baseline (Table 13). We exclude SIM-CoT from this comparison because Suleymanzade et al. (2026b) report weak SIM-CoT accuracy on the natural-language subset.

G EXPERIMENTAL DETAILS

This appendix includes the training and evaluation details in Section 4.1.

Latent prefix budget. Each of the K blocks holds c latent positions, so the prefix spans Kc positions: c=25 (150 positions) for the LLAMA backbones and c=13 (78 positions) for GPT-2. We use K=6, which covers the reference CoT step count for 99% of GSM8k-Aug examples. Longer problems fall back to autoregressive completion of the unsupervised tail.

Training curriculum. We follow the staged curriculum of Hao et al. (2025), converting one additional CoT step into a latent block every E_{stage} epochs. At epoch e, we use

$$K_e \;=\; \min(\lfloor e/E_{\text{stage}} \rfloor,\; K)\,,$$

latent blocks: up to the first K_e CoT steps become latent blocks, and the rest remain visible tokens. Thus e=0 is standard CoT fine-tuning and the curriculum saturates at K_e=K. We set the loop count at epoch e to $R_e = r\,K_e$, where r is the loops-per-block ratio. Since each pass refines all blocks in parallel, K_e controls the latent workspace width and R_e controls the depth. Unless stated otherwise, E_{stage}=1 and r=1, so the saturated model uses R=K=6. The loop-depth ablation (Section 4.3.2) fixes K=6 and varies r to decouple depth from width.

Optimization details. We initialize from each backbone's standard explicit CoT checkpoint, and fine-tune all parameters with AdamW and gradient-norm clipping at 1.0, in `bf16` for the Llama backbones and `fp32` for GPT-2. The learning rate is constant: $1{\times}10^{-4}$ for GPT-2 and LLAMA-3.2-1B-INSTRUCT, and $5{\times}10^{-5}$ for LLAMA-3.2-3B-INSTRUCT. The training objective (Equation (6)) weights the step loss at λ_{step}=0.05 (for both LOTUS and LOTUS-aux). Adding CODI weights its loss at 1.0. We train for 30 epochs with an overall batch size of 128, and select the best checkpoint by GSM8k validation accuracy, though the final epoch checkpoints perform similarly.

Natural-language training setup. For the natural-language stress-test on LOTUS, we keep the same 3B looped backbone but switch to the natural-language GSM8k-Aug training set (Wu et al., 2025), increase latent width to c=50 and set λ_{step} to 0.033. We initialize from the CoT checkpoint trained on the natural-language GSM8k-Aug. Other settings are identical to those used for the original GSM8k-Aug training.

LOTUS-aux auxiliary decoder. For LOTUS-aux, the auxiliary decoder is a full-size deep copy of the base model, initialized from its weights but trained together with the main model. This full-size choice matches SIM-CoT (Wei et al., 2025), whose auxiliary decoder is architecturally identical to the base model. We also tried lightweight auxiliary decoders but found them unstable in training.

Evaluation. Inference uses greedy decoding with batch size 1. This isolates per-example latency from batching effects and calculates the average latency per sample accurately. Latency is measured on a single NVIDIA H100 and decomposed into query, thought, and answer phases (Table 2).

Batched KV-cache sharing. For training, we left-pad shorter examples so the ⟨lat⟩ positions align across the batch. This lets us compute each example's question-prefix KV cache once and reuse it across all R looped iterations.

H PCL Versus Autoregressive NLL Lower Bounds

Here we provide the explanation mentioned in Appendix A.1. PCL (Appendix B) scores each gold token in parallel, $-\sum_{i,j} \log p_{\boldsymbol{\theta}}(T_{i,j} \mid \boldsymbol{h}^{(R)})$, whereas the autoregressive factorization also conditions on the preceding gold tokens, $-\sum_{i,j} \log p_{\boldsymbol{\theta}}(T_{i,j} \mid \boldsymbol{h}^{(R)}, T_{<(i,j)})$, where $T_{<(i,j)}$ denotes all gold tokens before position (i, j) in reading order. The lowest expected NLL each can reach is the corresponding conditional entropy, and since conditioning never increases entropy,

$$H\big(T_{i,j} \mid \boldsymbol{h}^{(R)}\big) \;\geqslant\; H\big(T_{i,j} \mid \boldsymbol{h}^{(R)}, T_{<(i,j)}\big), \tag{11}$$

with equality only when the gold tokens are conditionally independent given the latents. So PCL has a strictly higher NLL floor whenever cross-token dependence exists.

I Details for Latent Representation Analysis (Appendix A)

I.1 Greedy Readout Examples

Here we give examples of two correctly answered and two incorrectly answered GSM8K test problems. We show three readouts of the post-loop latents: **(A)** the per-block top-1 token sequence reading $\boldsymbol{h}^{(R)}$ through LOTUS's base LM head f_{head}; **(B)** the same readout applied to LOTUS-aux's post-loop latents $\boldsymbol{h}^{(R)}$ through its auxiliary head g_{head}; and **(C)** the autoregressive CoT produced by LOTUS-aux's trained auxiliary decoder from $\tilde{\boldsymbol{z}}^{(t)}$, conditioned on the same latents and decoded greedily from the start-of-CoT token. (A) and (B) show the top-1 token stream per block verbatim (newlines rendered as \n), truncated at 60 characters per block with . . . marking the cut. (C) shows the raw aux-decoder output. We additionally report the final answer the model generates autoregressively from $\boldsymbol{h}^{(R)}$ after all R=6 loops. In each **(A)/(B)** readout we list the K=6 latent blocks top to bottom, labeled B0–B5.

For readability, blue marks the readable gold content: well-formed «. . . » CoT steps and exact gold chain numbers (step operands and intermediate results), including a number stuttered back-to-back such as 540540 for 540. Wrong digit runs (e.g. 120120 or 3636) are left unmarked, and within each readout line we highlight only the first surfacing of each gold value.

Example 1 — Janet's ducks.

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? Reference: «16-3-4=9», «9*2=18». *Gold answer* 18.

(A) LOTUS LM head ($\boldsymbol{h}^{(R)}$ through f_{head}).

```
B0: <<3+4=7>>\n00>>\n7>>\n>>\n>>\n1616--9999999<<
B1: <<16-7=9>>\n00>>\n>>\n>>\n>>\n>>\n*2==1818>>\n>>\n>>\n1818<<
B2: <<9*2=18>>\n00>>\n>>\n>>\n18>>\n>>\n>>\n= 1818>>\n18>>\n>>\n...
B3: <<18*2=18>>\n>>\n>>\n>>\n>>\n>>\n>>\n>>\n>>\n>>\n=99>>\n>>\n18>>...
B4: ###134*7=>>\n18>>\n>>\n>>\n>>\n>>\n>>\n>>\n>>\n2=2 .>>\n181818>>...
B5: >>\n18=21818>>\n>>\n>>\n===.###2== ,>>\n1818222
```

(B) LOTUS-aux LM head ($\boldsymbol{h}^{(R)}$ through g_{head}).

```
B0: <<'+)7))<<1616--7= 1313\n\n###22*22
B1: =2189)#########2222=  1818#########22*9=
```

```
B2:  1818 per#########2922 =1818 fresh######99* no16= 18
B3: 18 fresh<|eot_id|>#########9* no no nothing=1818 Clean<|eot_...
B4: 18ly<|eot_id|>###### no nothing no nothing 1818<|eot_id|>###...
B5: ############ nothing nothing nothing=181818############### n...
```

(C) LOTUS-aux auxiliary decoder ($\tilde{z}^{(t)}$ through g_{head}, FR).

```
«3+4=7»\n«16-7=9»\n«9*2=18»\n«18=18»\n«9*2=18»\n«18+18=18»\n«18*2=18»\n
«18=18»
```

Final answer (autoregressive, after 6 loops). LOTUS: `### 18`. LOTUS-aux: `### 18`. Gold: 18. Both correct.

Reading the chains. (A): the gold intermediates **9** and **18** surface from B1 onward and stabilize by B5. (B): the same numerics $(9, 18, 16, 7)$ are present but interleaved with `<|eot_id|>` runs and filler. (C): the aux decoder produces the gold first two steps «3+4=7», «16-7=9», «9*2=18» and then loops on 18.

Example 2 — James's sprints.

Question: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week? Reference: «3*3=9», «9*60=540». *Gold answer* 540.

(A) LOTUS LM head ($\boldsymbol{h}^{(R)}$ through f_{head}).

```
B0: <<3*60=180>>\n00>>\n180180>>\n>>\n3==540540>>\n>>\n>>\n>>\n>...
B1: <<180*3=540>>\n00>>\n>>\n>>\n540>>\n###   540>>\n>>\n   >>\n...
B2: <<540*540=540>>\n00>>\n>>\n>>\n>>\n>>\n>>\n>>\n###  540>>\n>>\n ...
B3: <<540=180=540>>\n540>>\n>>\n>>\n>>\n>>\n>>\n>>\n###  540>>\n>>\n...
B4: ###540= =>>\n###>>\n>>\n>>\n>>\n 2 ######  540540>>\n>>\n   ###
B5: ### == >>\n#########    >>\n###    >>\n>>\n   >>\n
```

(B) LOTUS-aux LM head ($\boldsymbol{h}^{(R)}$ through g_{head}).

```
B0: 33*3= 180180180*###9993==  540540)######9
B1: 9*60= 540540))#########9*3 nothing = 540540)############
B2: /* nothing= 540540)#########540+### nothing  540############...
B3:  nothing is 540540################540,### also 540540540#####...
B4:  is540540540############540,### also 540540################...
B5: 540540################540### definitely 540540540############...
```

(C) LOTUS-aux auxiliary decoder ($\tilde{z}^{(t)}$ through g_{head}, FR).

```
«3*60=180»\n«180*3=540»\n«540=540»\n«540/60=9»\n«540=540»\n
«540/10=540»\n«540»
```

Final answer (autoregressive, after 6 loops). LOTUS: `### 540`. LOTUS-aux: `### 540`. Gold: 540. Both correct.

Reading the chains. (A): the gold step «3*60=180» appears in B0 already (a valid alternative decomposition: meters per day first), and **540** stabilizes from B1. (B): $180, 540, 9, 3$ are all present but with heavy `###` runs around them. (C): the aux decoder reproduces **180 → 540** cleanly, then loops.

Example 3 — Carla's download (failure).

Question: Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file? Reference: «200*40*.01=80», «80/2=40», «200/2=100», «40+100+20=160». *Gold answer* 160.

(A) LOTUS LM head ($\boldsymbol{h}^{(R)}$ through f_{head}).

```
B0: <<200/2=6=120>>\n6>>\n>>\n>>\n>>\n>>\n>>\n2>>\n505050>>\n605050<...
B1: <<40*.2=40>>\n120>>\n666>>\n>>\n>>\n>>\n>>\n>>\n60>>\n8080>>\n>>...
B2: <<60+40=120>>\n8>>\n>>\n>>\n>>\n>>\n>>\n>>\n>>\n6050>>\n>>\n>>\n...
B3: <<140+20=120>>\n8>>\n33>>\n>>\n>>\n>>\n>>\n>>\n2020=120>>\n>>\n>...
B4: <<140+20=120>>\n120>>\n33>>\n>>\n>>\n>>\n>>\n###+60+/1>>\n>>\n>>...
B5: 160160+20=120>>\n180>>\n>>\n>>\n>>\n>>\n180>>\n###+600+100120>>\...
```

(B) LOTUS-aux LM head ($\boldsymbol{h}^{(R)}$ through g_{head}).

```
B0: 100200/) 100100))<<100100**40 percentage,404040,<<100-+
B1: 2020= 80+ and###100100+20++ 120120)###100100+220+
B2:  120120 plus###100100*10020 140 plus###12080+2100= 120240)\n...
B3: ###120120+2020+120140140######120+202020 140140)\n######120+
B4: +2=  140140)\n)\n######120120+202020  140100)\n Ris######
B5: ###120+2020= 100100 Ris#########120+20= 100100 Ris##########...
```

(C) LOTUS-aux auxiliary decoder ($\tilde{z}^{(t)}$ through g_{head}, FR).
`«100+20=120»\n«120+20=140»\n«100+20=120»\n### 140`

Final answer (autoregressive, after 6 loops). LOTUS: `### 180`. LOTUS-aux: `### 120`. Gold: 160.

Reading the chains. The latents recover most of the right intermediates (**80**, **100**, **40**, **20**) but fail to combine them: (A) flickers between $120, 140, 160, 180$ in the late blocks; (B) commits to $120 + 20$ chains; (C) outputs `«100+20=120», «120+20=140», ...` and emits 140. Both methods land on plausible-but-wrong totals.

Example 4 — Melanie's vacuums (failure).

Question: Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with? Reference: `«5*2=10», «10+2=12»`. *Gold answer* 18. *One extra step is omitted from the reference; the true chain is* $5 \to 10 \to 12 \to 18$ *(*$12 \times 3/2 = 18$*).*

(A) LOTUS LM head ($\boldsymbol{h}^{(R)}$ through f_{head}).

```
B0: <<5*2=10>>\n33>>\n12>>\n12>>\n22=1312>>\n1212121313<<
B1: <<10+2=17>>\n5>>\n5>>\n17>>\n>>\n1//1181818.55<<
B2: <<9/(1/2/5=20.5>>\n20>>\n15.18>>\n>>\n>>\n>>\n5>>\n<<
B3: <<15*1=30>>\n15>>\n5>>\n5>>\n20>>\n5=151818>>\n>>\n55<<
B4: <<3=15>>\n30>>\n15>>\n5>>\n5>>\n>>\n>>\n55515>>\n15>>\n5>>\n...
B5: <<15+5=15>>\n15>>\n5>>\n>>\n>>\n17>>\n55515>>\n18>>\n18>>\n>...
```

(B) LOTUS-aux LM head ($\boldsymbol{h}^{(R)}$ through g_{head}).

```
B0: 105* half 1010)+101010+2)= 1212)<<1212*3
B1: 3)1512..<<121212*3 third = 3636.###121212*3)
B2:   3636)######12+33= 3636)######36+125=
B3: 3636/##########36+12++  4836 Egg#########36+12+=
B4: 483636##########+5== 4836#############36+5  3636###
B5: ############15+3  3636################15###1 = 363615###
```

(C) LOTUS-aux auxiliary decoder ($\tilde{z}^{(t)}$ through g_{head}, FR).
`«10+2=12»\n«10+2=12»\n«10+2=12»\n«12+2=14»\n...`

Final answer (autoregressive, after 6 loops). LOTUS: `### 30`. LOTUS-aux: `### 36`. Gold: 18.

Reading the chains. (A): LOTUS's latents flicker between $15, 18, 30$ across blocks. The valid sub-chain $\mathbf{5 \to 10 \to 12 \to 18}$ surfaces in B5, but the final readout commits to 30. (B): LOTUS-aux locks onto a wrong $\times 3$ branch (presumably "a third" read as multiply-by-three), with 36 saturating B4 to B5. (C): the aux decoder loops on `«10+2=12»`.

What these readouts are (and are not). The readouts (A), (B), (C) are just diagnostic projections of the post-loop continuous latents back into the discrete token space. They are not part of the inference pipeline. At inference, the final answer is generated autoregressively from the continuous post-loop latents (the "Final answer" line in each example), conditioning on the latent representation itself rather than on any discretized chain. The intermediate streams are intended to give intuition about *what numbers the latents place mass on*, not as a quality measure of the model's reasoning: a noisy or repetitive (A)/(B) stream is compatible with a correct final answer, and a clean (C) chain is not required for the model to be correct.

I.2 Details for Multi-Path Analysis (Appendix A.2)

Datasets. Note that GSM8K-Aug (Deng et al., 2023) augments the original GSM8K (Cobbe et al., 2021) training split (about 7.5K problems) into the 385,620 training examples we use. GSM8K is thus a subset of GSM8K-Aug, and the multi-path bank below is built from the *original* GSM8K problems, which are also in the training set.

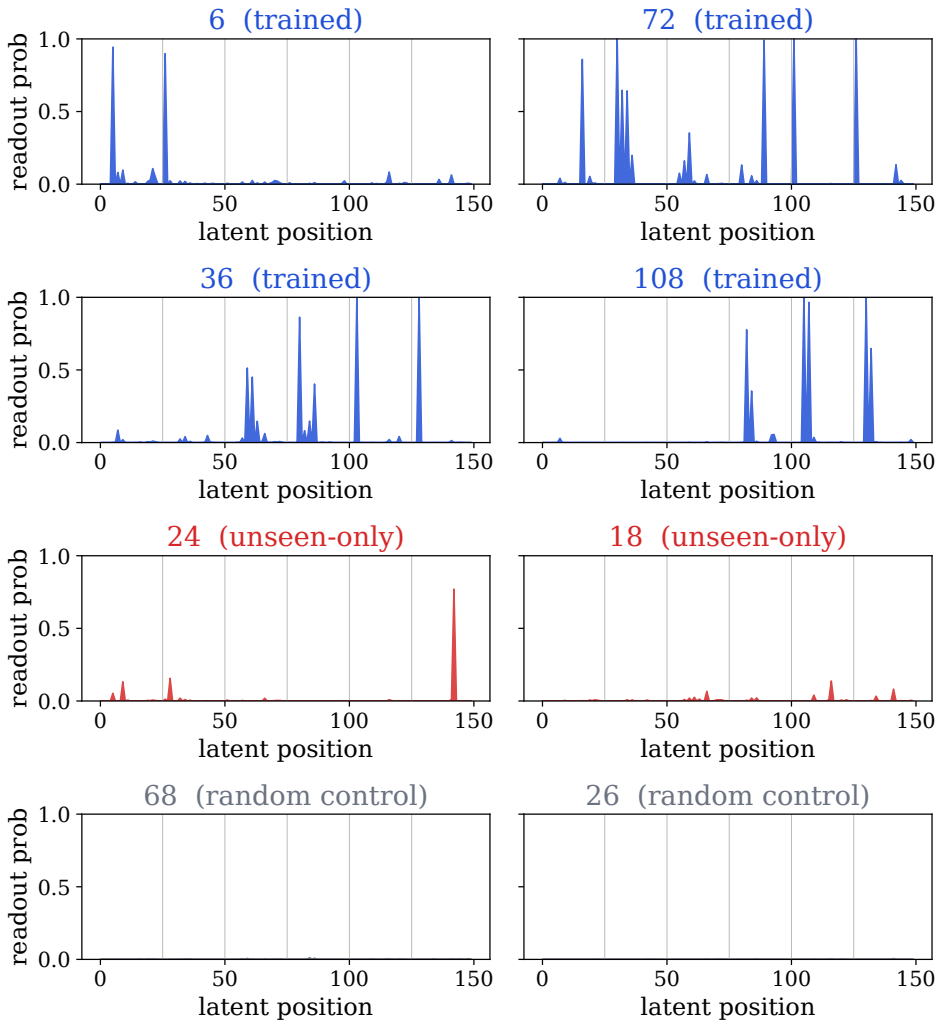


Figure 4: Multi-path readout probability for the example in Appendix I.3. The only-U intermediates do not appear in the trained path, question, or answer, but they surface in the post-loop latent readout.

Path bank. For each question we identify its trained path (its gold chain in the training data) and an unseen-but-valid alternative drawn from a bank of up to ten correct paths obtained by rejection-sampling LLAMA-3.1-8B (Grattafiori et al., 2024) on the original GSM8K training questions (7,444 questions with 58,155 verified paths in total), with each alternative verified absent from training.

Candidate sets and filtering. We partition intermediate numbers into three disjoint sets: G (trained-path only), U (unseen-path only), and a random control from other questions matched to U in both cardinality and single/multi-token composition. Numbers shared by both paths or appearing in the question or answer are excluded, since they cannot indicate which chain a readout recovered. The random control is fixed and shared across all four readouts and verified disjoint from the trained path, unseen path, question, and answer numbers. Intersecting the bank with our training set and keeping questions with $\geqslant 2$ disjoint intermediates per side yields 653 candidate questions. We then apply a token-clean filter, requiring every only-U number to share no sub-token with any trained-path, question, or answer number, leaving the final 341 questions, spanning 871 G, 974 U, and 974 Random numbers. Random stays matched to U in cardinality and composition under every criterion.

Metrics. We score a target number by its best-position, per-token NLL. A number spans one or more consecutive sub-tokens, and at every latent position the readout gives a softmax distribution

over the vocabulary. We slide the number's sub-tokens across the latent positions. At each placement we average the sub-tokens' NLLs at their aligned positions, and keep the lowest such per-token average over all placements (for a single-token number, simply the negative log of its highest readout probability across positions). Per-token normalization puts single and multi-token numbers on the same scale, and scoring only the most probable span avoids double counting overlaps. We average this over the numbers in sets G, U, and Random within each question, then over the 341 questions. The $P(U \in \text{top-}k)$ columns instead report a hit rate: the fraction of U numbers recovered within the top-k candidates (top-k membership at any position for single-token numbers, and a consecutive top-k match for multi-token ones), counting each number once however many positions it surfaces at, averaged within each question and then over the 341 questions.

I.3 Multi-Path Readout Example

This example (referenced from Appendix A.2) shows a question that admits two valid reasoning chains with the same answer 108, read out by LOTUS ($\boldsymbol{h}^{(R)}$ through the base LM head f_{head}, i.e. the first row of Table 9).

> *Q:* The bus driver drives an average of 2 hours each day, 5 days a week. From Monday to Wednesday he drove at an average speed of 12 kilometers per hour, and from Thursday to Friday at an average speed of 9 kilometers per hour. How many kilometers did the driver travel during these 5 days?
>
> *Trained path G:* `3*2=6 -> 6*12=72 -> 2*2=4 -> 4*9=36 -> 72+36=108`.
> *Unseen path U:* `2*12=24 -> 24+24+24=72 -> 2*9=18 -> 18+18=36 -> 72+36=108`.
> $\text{only}_G = \{3, 4, 6\}$, $\text{only}_U = \{24, 18\}$ (question and final answer excluded).

Figure 4 visualizes the same example, aligning the trained path, the unseen valid path, and the latent readout positions where the only-U intermediates surface. The only-U intermediates 24 and 18 are the per-day distances in the alternative decomposition, which forms daily distances before summing rather than aggregating hours first. Although neither appears in the trained reference path, both surface in the post-loop latent readout through the base LM head: $P(\texttt{24})$=0.77 (top-1, position 142) and $P(\texttt{18})$=0.14 (top-2, position 116). Every number here is a single token, so no only-U number shares a sub-token with any trained-path, question, or answer number, confirming the surfacing is genuine unseen-chain content rather than a digit-overlap artifact.