SUPERPOSITION AND DISTRIBUTED MEMORY IN CONTINUOUS CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous chain-of-thought reasoning feeds hidden states back into a language model while retaining a persistent key–value (KV) history. In this work, we observe a distinction between latent generation and answer readout in Coconut: restricting latent-history access to recent positions during generation preserves most answers, whereas restricting it to the final position at readout changes many answers. Restoring earlier cached values also recovers much of the lost accuracy after history corruption while the final latent input remains fixed. To understand how this historical information can support reasoning, we develop a frontier–history model for graph reachability. We show that a two-block Transformer can implement exact frontier updates and answer readout by superposing the current frontier in feedback and retaining earlier frontiers in the KV cache. Further experiments on a trained two-layer Coconut model reveal depth-aligned feedback, depth-selective historical readout, and attenuation of revisited nodes. However, paired interventions show that feedback also retains update-relevant history beyond the current frontier. These results offer a distributed account of continuous reasoning: superposition need not entail cumulative storage in every feedback vector.

1 INTRODUCTION

Chain-of-thought (CoT) reasoning improves language models' performance on multi-step problems (Wei et al., 2022), but requires autoregressive generation of discrete intermediate tokens. Chain of Continuous Thought (Coconut) instead feeds hidden states back into the model as continuous thoughts (Hao et al., 2025). Subsequent methods, including CODI, SIM-CoT, and LOTUS, advance latent reasoning through self-distillation, supervised latent objectives, and looped architectures (Shen et al., 2025; Wei et al., 2025; Fan et al., 2026). Despite this progress, how latent states support reasoning remains unclear. In Coconut, each step produces both a feedback vector for the next step and layerwise entries in the key–value (KV) cache. Later computation can access information through recurrent feedback and attention to earlier positions. We study how reasoning information is distributed across these two routes.

Existing mechanistic studies use probing and activation patching to trace information across latent steps. They identify decodable intermediate variables and bridge representations, but also report shortcut-like behavior on longer reasoning chains (Liang & Pan, 2026; Li et al., 2026). Decodability alone does not establish causal use: similar patterns can arise in matched controls, and their interpretation can depend on the training regime (Aswal et al., 2026). Cache-level interventions provide complementary evidence that cached values can influence answer prediction (Ding et al., 2026). Connecting these findings requires an account of both what the current feedback vector represents and how earlier cached states contribute to subsequent computation and readout.

A prevalent theoretical account explains continuous reasoning through superposition. In the graph-reachability construction of Zhu et al. (2025), each continuous thought represents a normalized sum of *all nodes reached so far*, and the final thought supports answer readout. Subsequent work studies the learning dynamics of this mechanism (Zhu et al., 2026), while recent capacity analyses compare frontier and cumulative superpositions within a single vector (Gu et al., 2026). These results establish how cumulative representations can support reasoning, and they leave open a complementary question: *can a continuous reasoner preserve earlier information in cached history without carrying all of it forward in each feedback vector?*

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.

We begin with interventions on Coconut models across arithmetic and logical reasoning tasks. Both feedback and historical access contribute to performance, with effects that depend on the task and intervention stage. In particular, earlier latent positions remain useful after latent computation has finished, for instance, on GPT-2 Coconut/GSM8K, restricting answer readout to the final latent position changes 65.0% of clean answers and reduces exact-match accuracy by 17.89%. These interventions characterize the trained readout's use of history, and motivate a mechanism in which earlier information remains accessible through historical positions as the feedback vector evolves.

To make this possibility concrete, we develop a *frontier–history model*. In this construction, the feedback vector superposes the current search frontier, namely the nodes first reached at the current depth, while cached records preserve earlier frontiers. As search advances, a node can leave the feedback vector yet remain available through its earlier cache entry. The model uses these records to exclude previously visited nodes during expansion and identify the reachable candidate at readout. In contrast to the cumulative construction of Zhu et al. (2025), this mechanism supports reasoning through superposition without carrying every previously reached node in the current feedback vector.

Our contributions are threefold:

- We characterize the roles of recurrent feedback and cached history through interventions during latent generation and answer readout, showing that earlier latent positions can continue to support answers after the trajectory is complete.
- We construct an explicit two-block Transformer that implements exact frontier updates, recovers the visited set from retained frontier records, and returns the correct reachable candidate (Theorem 1). This establishes that distributed storage suffices for the computation, without cumulative storage in the feedback vector.
- We identify depth-selective historical readout in a trained two-layer Coconut model. With the latent trajectory fixed, earlier cache positions support answers at corresponding depths, while feedback preferentially represents the current frontier and retains additional update-relevant context. These findings connect the evolution of feedback representations with the continued use of earlier cached states, revealing how reasoning information remains accessible across positions.

2 Motivating Empirical Observations

We begin by asking whether continuous reasoning is summarized into the current latent state, or whether earlier latent states remain independently useful. We use GPT-2 Coconut (Hao et al., 2025) on GSM8K (Cobbe et al., 2021) as our primary setting and report broader cross-model and cross-dataset results in Appendix B.

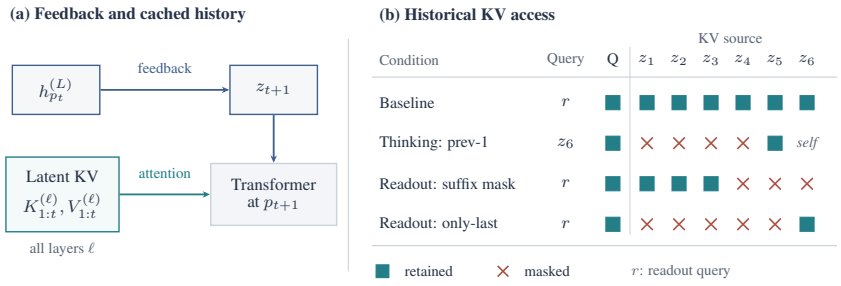

Figure 1: Two information routes in continuous reasoning. Recurrent feedback passes the current hidden state to the next latent position, while earlier keys and values remain accessible through the KV cache. Readout interventions leave the generated trajectory unchanged and modify only its subsequent accessibility.

Feedback and persistent history. Let z_t be the input at latent position p_t and $h_{p_t}^{(L)}$ its final-layer output. Coconut recurrently feeds this output back as $z_{t+1} = h_{p_t}^{(L)}$. In parallel, every layer stores a key $K_t^{(\ell)}$ and value $V_t^{(\ell)}$, so earlier latent positions remain accessible through attention. We refer to these two routes as *recurrent feedback* and *persistent KV history*, respectively.

Prior work has studied the causal organization of latent reasoning states (Liang & Pan, 2026; Li et al., 2026) and recently identified KV-cache components as causal carriers in latent-CoT models (Ding et al., 2026). Here we focus on whether earlier cached states contribute beyond the terminal latent state used by the trained readout.

Evaluation protocol. We use greedy decoding and report exact-match accuracy (Acc.), $\Delta\mathrm{Acc}$ relative to the corresponding clean run, and AnsMatch, the percentage of answers unchanged from that run. As illustrated in Figure 1, readout interventions first generate the latent trajectory normally and then restrict access to selected latent KV entries during answer generation; question KV remains visible. Restoration replaces earlier latent inputs with those from a donor example while retaining the target's final latent input, then restores selected target KV entries from the clean run. Thinking prev-1 instead regenerates the latent trajectory with recurrent feedback and a local history window, followed by unrestricted readout.

Observation 1: History dependence differs between thinking and readout. We first ask whether persistent history is used similarly during latent generation and answer readout.

During thinking, Thinking prev-1 retains recurrent feedback while restricting historical access to a local window, after which the regenerated trajectory is read out without restriction. From Table 1, we suggest that recurrent feedback and local history therefore support much of the latent computation, while the final reader depends more strongly on persistent earlier states.

Condition	Acc.	ΔAcc.	AnsMatch
Clean	34.42	0.00	100.00
Thinking prev-1	31.46	-2.96	70.51
Readout last-one	16.53	-17.89	35.03
Readout none	5.69	-28.73	11.52

Table 1: Phase-specific history interventions for GPT-2 Coconut on GSM8K ($n = 1319$).

Observation 2: Answer-relevant information is distributed across the latent history. We next ask whether readout dependence is concentrated at a particular latent position. Masking any single position has a substantially smaller effect than removing several positions together (Table 2). These results suggest that no single latent position alone accounts for the observed readout dependence. Although the suffix trend is not strictly monotonic, the joint effect of removing multiple positions is substantially larger than that of any single-position intervention, supporting distributed readout dependence.

	Clean	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
Mask only z_i	34.42	34.42	29.11	34.12	27.45	34.50	31.84
Mask suffix $z_{i:6}$	34.42	5.69	8.34	18.27	20.32	30.48	31.84

Table 2: Readout masking in GPT-2 Coconut on GSM8K ($n = 1319$). The first row removes only z_i; the second removes the complete suffix $z_i, \ldots, z_6$.

Observation 3: Earlier cached history contributes beyond the final recurrent input. The preceding masking results establish that earlier cache entries affect readout, but do not separate their contribution from the final latent input produced using that history. We therefore corrupt the earlier latent inputs while retaining the target's final latent input. Thus, earlier cached history makes an answer-relevant causal contribution that is not reducible to the retained final recurrent input or final cached position. The stronger recovery from values is consistent with recent cache-level causal evidence for value trajectories in latent reasoning (Ding et al., 2026).

	Clean	Corrupted	Final KV	Earlier KV	Earlier K	Earlier V
Acc.	34.42	5.23	8.18	33.81	6.29	24.87

Table 3: Within-model KV restoration for GPT-2 Coconut on GSM8K ($n = 1319$). Corrupted replaces the target's earlier latent inputs with a donor's while retaining the target's final latent input. Each restoration starts independently from this corrupted run and restores the indicated target cache entries from the clean run.

In the graph-reachability construction of Zhu et al. (2025), each continuous thought superposes all nodes reached so far, and the final thought supports answer readout. This establishes that cumulative storage suffices for the task, but does not explain the trained reader's dependence on earlier history observed above. Our interventions establish the use of earlier cached information, not its absence from the final vector. We therefore ask whether superposition can support reasoning without accumulating all previously reached nodes in every feedback vector. Section 3 studies this possibility in graph reachability: feedback represents the current frontier, while cached states retain earlier frontiers for subsequent computation and readout.

3 Frontier–History Model of Continuous Reasoning

We now make the separation suggested by Section 2 explicit in a graph-search model. The construction below implements this division with two Transformer blocks.

3.1 Graph Reachability and the Recurrent Interface

Graph task and search frontiers. Following Zhu et al. (2025), we consider a directed graph $G = (V, \mathcal{E})$ with $N \geq 2$ nodes, a root $r \in V$, and two distinct candidate nodes $c_1, c_2 \in V$. Exactly one candidate is reachable from r, which is denoted by $c_\star$. The input lists all graph edges as M pairs (s_m, t_m), $m \in [M]$, where s_m and t_m are the source and target. We fix a nonnegative integer bound $M_{\max}$ and allow $0 \leq M \leq M_{\max}$. Breadth-first search (BFS) groups nodes by their shortest-path distance from r. Let $\mathrm{dist}(r, v)$ be this distance, with value $+\infty$ when v is unreachable. At each integer depth $c \geq 0$, define $F_c = \{v \in V : \mathrm{dist}(r, v) = c\}$ and $R_c = \bigcup_{j=0}^{c} F_j$. The *frontier* F_c contains the nodes first reached at depth c; the *visited set* R_c also includes all earlier frontiers. For a node set $S \subseteq V$, its outgoing neighbors are $\mathcal{N}(S) = \{v \in V : \exists u \in S,\ (u, v) \in \mathcal{E}\}$. In our construction, one BFS step expands the current frontier and removes visited nodes, namely

$$F_0 = R_0 = \{r\}, \qquad F_{c+1} = \underbrace{\mathcal{N}(F_c)}_{\text{outgoing neighbors}} \setminus \underbrace{R_c}_{\text{already visited}} . \tag{1}$$

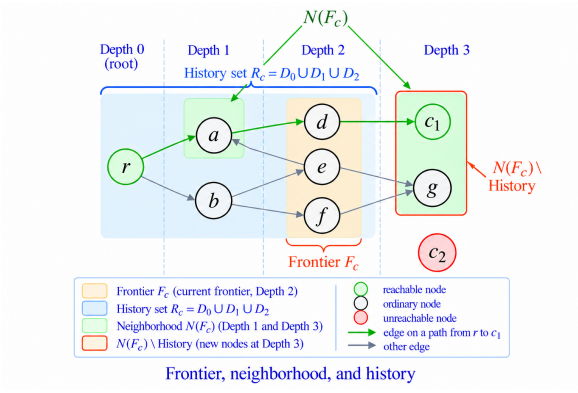

Figure 2: One frontier update in graph reachability. At depth 2, $F_2 = \{d, e, f\}$ and $R_2 = \{r, a, b, d, e, f\}$. Expansion gives $\mathcal{N}(F_2) = \{a, c_1, g\}$. Node a is encountered again through $e \to a$, but is already visited, so $F_3 = \{c_1, g\}$.

Vector for the current frontier. Give each node v its own unit coordinate vector e_v, and reserve one additional unit coordinate $e_\star$ as an *anchor*. All these vectors are mutually orthogonal. For a set $F \subseteq V$, encode its nodes by the normalized vector $\phi(F) = \frac{e_\star + \sum_{v \in F} e_v}{\sqrt{1+|F|}}$. Thus each node in F has the same positive coordinate and every other node has coordinate zero. For example, the frontier in Figure 2 is encoded by $\phi(F_2) = (e_\star + e_d + e_e + e_f)/2$. The anchor ensures $\|\phi(F)\|_2 = 1$ even for an empty frontier, i.e. $\phi(\varnothing) = e_\star$. The feedback vector will encode F_c, and the visited set R_c will be recovered from the current frontier record together with cached earlier records.

Note that the setting here differs from that of Zhu et al. (2025), where each latent thought sums over all nodes reachable within c steps, corresponding to $R_c = \bigcup_{j=0}^{c} F_j$. Here, the feedback carries only the current frontier F_c, with earlier frontiers retained in cached records.

Prompt and two-block Transformer. Following (Zhu et al., 2025, Figure 1), the prompt starts with $\langle s \rangle$, uses $\langle e \rangle$ to end each edge, and encloses the two candidates between $\langle Q \rangle$ and $\langle R \rangle$. The root is the last prompt token:

$$\langle s \rangle, s_1, t_1, \langle e \rangle, \ldots, s_M, t_M, \langle e \rangle, \langle Q \rangle, c_1, c_2, \langle R \rangle, r. \tag{2}$$

The full prompt sequence length is bounded by $P_{\max} = 3M_{\max} + N + 6$.

We use a two-layer causal Transformer of width d, with three softmax attention heads and a positionwise FFN in each layer. Each FFN has one ReLU hidden layer and biases, and all parameters are shared across positions. At position i, the initial state is $h_i^{(0)} = x_i + p_i$, where x_i is the content embedding and p_i is a sinusoidal positional embedding. Let MHA_ℓ and FFN_ℓ denote the attention and feed-forward maps, for $\ell = 1, 2$, the updates are

$$u_i^{(\ell)} = h_i^{(\ell-1)} + \mathrm{MHA}_\ell\big(h_{\le i}^{(\ell-1)}\big), \qquad h_i^{(\ell)} = \mathrm{Norm}_2\Big(u_i^{(\ell)} + \mathrm{FFN}_\ell(u_i^{(\ell)})\Big), \tag{3}$$

where $\mathrm{Norm}_2(y) = y/\|y\|_2$ is Euclidean normalization, applied only after the FFN residual addition. Attention includes every position $j \le i$, including the current position, with no additional masking. The construction ensures that all normalization inputs are nonzero.

3.2 Exact Frontier Updates and Answer Readout

Write z_c for the input at frontier position i_c and i_A for the answer position.

Theorem 1. *For any integers $N \ge 2$ and $M_{\max} \ge 0$, there exists a two-block Transformer with the architecture above and parameters depending only on $N, M_{\max}$ such that every legal prompt, initialized with $z_0 = e_r$ and run for $N - 1$ additional feedback steps before $\langle A \rangle$, satisfies:*

i. ***Frontier updates.*** *For $0 \le c < N$, $h_{i_c}^{(2)} = \phi(F_{c+1})$. Feeding each output into the next step gives $z_c = \phi(F_c)$ for $1 \le c < N$.*

ii. ***Visited-set recovery.*** *Block 1 retains z_c in frontier coordinates; a uniform causal head in Block 2 returns $b_c \in \mathbb{R}^N$ with $\mathrm{supp}(b_c) = R_c$, where* supp *denotes the nonzero node coordinates.*

iii. ***Answer readout.*** *The answer output is $h_{i_A}^{(2)} = e_{c_\star}$, giving the reachable candidate a unit logit margin over every other node.*

Block 1: Prompt Parsing and Graph Records. The prompt in Equation equation 2 lists each edge's source and target as separate tokens. Block 1 collects these tokens into one edge record, groups the two answer candidates into a candidate record, and preserves the recurrent input as a frontier record.

Figure 3 shows where these records are stored within the hidden state. At an edge delimiter $\langle e \rangle$, the coordinate blocks S and T store the edge's source and target. At the candidate delimiter $\langle R \rangle$, the block C stores the two candidates. At a root or latent position, the block U stores the current frontier vector, including its anchor coordinate when present. The role coordinates identify which kind of record each position carries. The other graph-record blocks at that position are zero. The same layout also reserves X for input content, π for positional encoding, and a work block for temporary attention outputs. These blocks are used while computing the records. The Block 1 FFN, together with its residual connection, clears them after use, so the output $h_i^{(1)}$ retains only the structured records and role coordinates for Block 2. We first describe the two attention heads that collect the preceding node tokens.

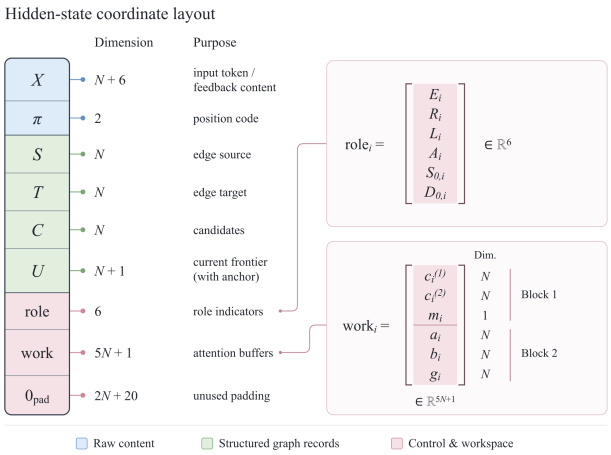

Figure 3: Hidden-state coordinate layout. The role entries $E_i, R_i, L_i, A_i, S_{0,i}, D_{0,i}$ denote edge delimiters, candidate delimiters, root/frontier positions, answer positions, the initial sink, and other positions, respectively, and are one-hot encoded. In the work block, $c_i^{(1)}, c_i^{(2)}$ store approximate node copies from $i-1, i-2$, while m_i is the Block 1 phase marker; a_i, b_i, g_i are the expansion, history, and candidate-head outputs in Block 2.

Copy head. The input to the first block at position i is $h_i^{(0)} = \begin{bmatrix} X_i \\ \pi_i \\ \mathbf{0}_{d-N-8} \end{bmatrix} \in \mathbb{R}^d$, where $X_i \in \mathbb{R}^{N+6}$ contains the node, anchor, and special-token coordinates, $\pi_i \in \mathbb{R}^2$ is the positional encoding, and the remaining coordinates are initialized to zero.

For each offset $\ell \in \{1, 2\}$, we choose the projection matrices

$$Q^{(\ell)} = \begin{bmatrix} \mathbf{0}_{2\times N} & \mathbf{0}_{2\times 6} & I_2 & \mathbf{0}_{2\times(d-N-8)} \end{bmatrix},$$
$$K^{(\ell)} = \begin{bmatrix} \mathbf{0}_{2\times N} & \mathbf{0}_{2\times 6} & \eta\,\mathrm{Rot}_\ell & \mathbf{0}_{2\times(d-N-8)} \end{bmatrix},$$
$$V^{(\ell)} = \begin{bmatrix} I_N & \mathbf{0}_{N\times 6} & \mathbf{0}_{N\times 2} & \mathbf{0}_{N\times(d-N-8)} \end{bmatrix},$$

where $\eta > 0$ controls attention concentration, and Rot_ℓ rotates the positional encoding by $\ell\omega$, so that $\mathrm{Rot}_\ell\, \pi_j = \pi_{j+\ell}$, with $\pi_j = (\cos(j\omega), \sin(j\omega))^\top$ and $\omega = 1/(P_{max}+1)$.

The query and key projections extract positional information for matching, while the value projection extracts node information, i.e. $q_i^{(\ell)} = Q^{(\ell)} h_i^{(0)} = \pi_i$, $k_j^{(\ell)} = K^{(\ell)} h_j^{(0)} = \eta\,\mathrm{Rot}_\ell\, \pi_j$, and $v_j^{(\ell)} = V^{(\ell)} h_j^{(0)} = X_{j,[N]}$. Thus, the attention score from position i to a visible position $j \le i$ is

$$\left\langle q_i^{(\ell)}, k_j^{(\ell)} \right\rangle = \eta \left\langle \pi_i, \mathrm{Rot}_\ell\, \pi_j \right\rangle = \eta \left\langle \pi_i, \pi_{j+\ell} \right\rangle = \eta \cos\big((i-j-\ell)\omega\big). \tag{4}$$

For $\ell < i \le P$, this score is uniquely maximized at $j = i - \ell$. Choosing a sufficiently large finite η therefore concentrates attention on the intended predecessor. The resulting approximate copies are written into the $c^{(1)}$ and $c^{(2)}$ work sub-blocks of $u_i^{(1)}$ respectively.

Marker head. The marker head indicates whether the reasoning delimiter $\langle R \rangle$ is visible under causal attention. With zero queries and keys and values $X_{j,R}$, where $X_{j,R}$ flags the unique delimiter $\langle R \rangle$, uniform causal attention writes $m_i = \frac{1}{i}\sum_{j=1}^{i} X_{j,R}$ into the work block, so $m_i = 1/i$ if $\langle R \rangle$ is visible and $m_i = 0$ otherwise.

Shared FFN. The shared ReLU FFN uses fixed thresholds to convert the approximate predecessor copies into exact node indicators. Before normalization, it writes $S_i = e_{s_m}$ and $T_i = e_{t_m}$ at the delimiter ending edge (s_m, t_m), and $C_i = e_{c_1} + e_{c_2}$ at $\langle R \rangle$. At root or latent positions, a gate controlled by the phase marker m_i and the special-token flags preserves the current node-and-anchor input in U_i. At each position, unused record blocks remain zero, and the FFN writes the corresponding role coordinates. The full construction is given in Appendix C.3.

Block 2: Frontier Updates and Answer Readout. Block 1 converts the serialized prompt and recurrent input into canonical records for graph edges, the two answer candidates, and the current frontier. Block 2 combines the graph records with the frontier history. As illustrated in Figure 4. Its three attention heads perform frontier expansion, visited-set recovery, and candidate retrieval. A shared ReLU FFN converts their outputs into the next frontier or the answer. Projection matrices, finite-softmax margins, the FFN realization, and a flowchart of Block 2 are given in Appendix C.4.

Let $P_u = [I_N\ 0]$ project a frontier vector onto its node coordinates. At frontier position i_c, write $a_c, b_c \in \mathbb{R}^N$ for the expansion- and history-head outputs. At the answer position i_A, write $g_A \in \mathbb{R}^N$ for the candidate-head output.

Expansion head. The expansion head compares the node coordinates of z_c with the source of each edge record and returns a weighted sum of its target coordinates. Finite softmax attention gives nonmatching edges nonzero weight, but the construction separates neighbors from non-neighbors: for constants $0 \le \mu_- < \mu_+$ fixed by N and $M_{\max}$,

$$(a_c)_v \ge \mu_+ \quad \text{if } v \in \mathcal{N}(F_c), \qquad \text{and } (a_c)_v \le \mu_- \quad \text{if } v \notin \mathcal{N}(F_c).$$

History head. With zero queries and keys, this head attends uniformly to all visible positions. Only frontier records have nonzero history values. If D_c positions are visible at i_c, its output is $b_c = \frac{1}{D_c}\sum_{s=0}^{c} P_u z_s$, and $\mathrm{supp}(b_c) = \bigcup_{s=0}^{c} F_s = R_c$ Thus feedback carries the current frontier, while the current and cached frontier records together recover the visited set.

Candidate head. At the answer position, this head gives the candidate record a higher score than every other visible position. With the finite scale chosen in Appendix C.4.3, the head produces

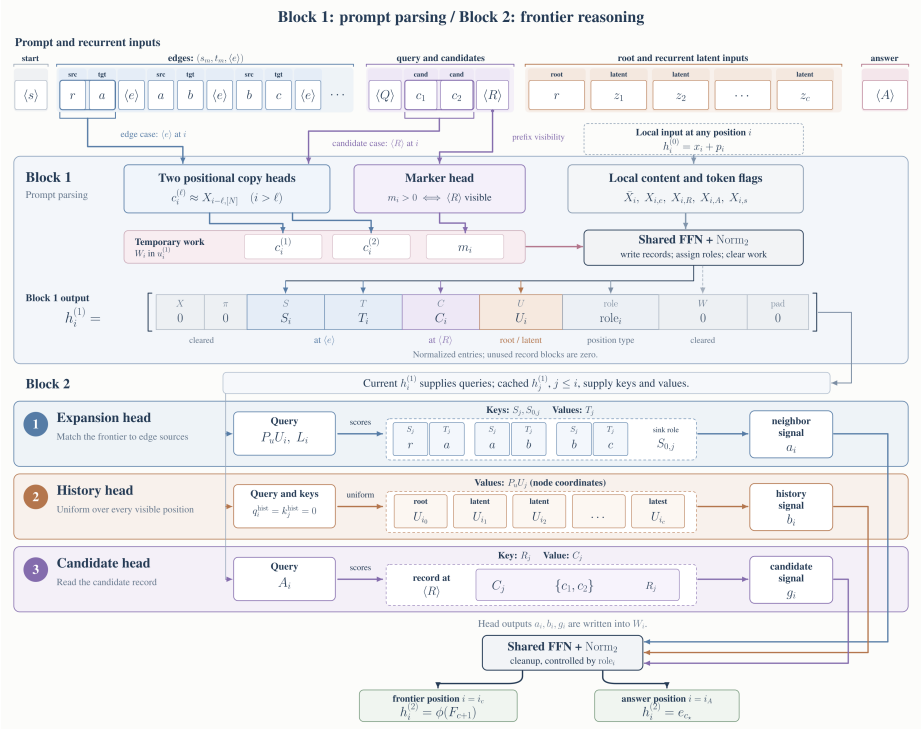

Figure 4: Block 1 constructs graph records; Block 2 updates the frontier and reads out the answer.

$g_A = \gamma_A(e_{c_1} + e_{c_2})$, where $\gamma_A \approx 1$. The candidate signal identifies the admissible outputs, while the history signal identifies which candidate is reachable.

Shared FFN. At a frontier position i_c, the FFN suppresses visited nodes in the expansion signal and returns $h_{i_c}^{(2)} = \phi(F_{c+1})$. At i_A, it combines the candidate and history signals to return $h_{i_A}^{(2)} = e_{c_\star}$.

The empirical observations suggest that recurrent feedback and cached history play distinct roles, but do not specify what each represents. Our construction uses exact FFN residual cleanup to make one such separation concrete: feedback carries the current frontier in superposition, while cached records preserve earlier frontiers for visited-set recovery and answer readout. This proves that graph search need not accumulate all visited nodes in each feedback vector; it does not require a trained Transformer to erase every history-dependent residual. We next distinguish frontier-selective representations and historical readout from the stronger causal prediction that cached frontier records alone control visited-node exclusion.

4 Testing the Frontier–History Mechanism

The history dependence observed in Section 2 motivates the frontier–history mechanism in Theorem 1. We test three implications in trained models: whether feedback tracks the current frontier, whether earlier frontiers remain accessible through the cache, and whether previously visited nodes are attenuated. We then use paired interventions to distinguish these representational signatures from the causal update circuit in our construction.

Evaluation setting. We train GPT-2-style Coconut decoders with two and six Transformer layers from scratch on simplified ProsQA, using numerical node identifiers and a curriculum with up to four continuous thoughts. The models are trained by AdamW, and a constant learning rate of 1×10^{-4}, without a learning rate warmup or decay schedule. We use the graph-query format defined above and focus on the two-layer results. Training details and native-task baselines are deferred to Appendix D, and the six-layer comparison appears in Appendix E.

Let P_t denote the t-th latent position after the root and z_t its input. The root position produces z_1 for P_1. We distinguish this feedback input from the layerwise KV entries written while processing it.

4.1 Depth-Aligned Feedback and Historical Readout

We test representation and readout separately. On unmodified questions, we rank nodes by their cosine similarity to z_t and measure separation of each BFS frontier from other reachable depths using the area under the AUROC. For readout, we evaluate target depths 1–4: each query generates four latent steps normally before the answer's attention is restricted to one latent position across all layers. Prompt KV and the answer position remain fixed. This extends the readout intervention in Section 2 to identify which depth each cache position makes available.

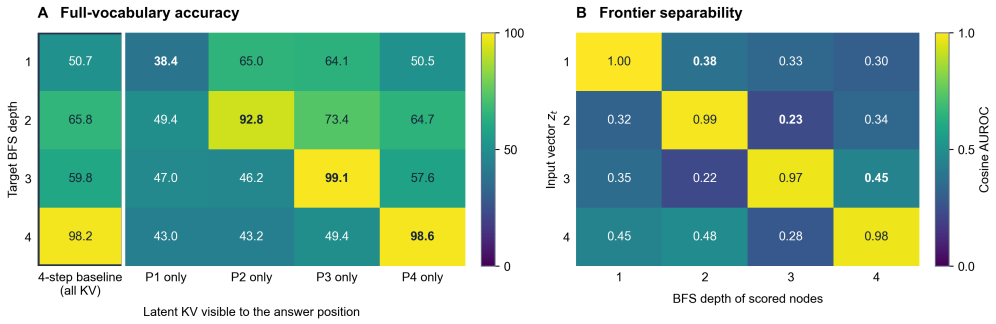

Figure 5: **Frontier information in feedback and cached history.** (A) Full-vocabulary answer accuracy (%) for targets at BFS depths 1–4, with the root at depth 0. All conditions use four latent steps; "P_t only" exposes that position's latent KV to the answer, while retaining prompt KV. (B) Cosine-score AUROC of z_t for nodes at each depth against nodes at other reachable depths, measured on unmodified questions.See Appendix D for details on query construction.

Feedback tracks the current depth. The cosine scores show a clear diagonal pattern (Figure 5B): $z_1, \ldots, z_4$ separate their corresponding frontiers with AUROCs of 1.00, 0.99, 0.97, and 0.98, respectively. Feedback therefore preferentially represents the corresponding frontier at each step. Appendix F further shows a shift from F_t at the input toward F_{t+1} at the output for P_1–P_3, consistent with the frontier update in Theorem 1.

Earlier cache positions support depth-selective readout. Depth-matched cache positions remain sufficient for answering earlier-depth queries even after all four reasoning steps (Figure 5A). In some cases, selective readout even outperforms access to the full cache, suggesting that relevant history remains available but is not always used effectively by the unrestricted reader.

Depth 1 is an informative exception: despite z_1's AUROC of 1.00, P_1-only readout achieves only 38.4% accuracy. As in **Observation 2** of Section 2, separability in a representation does not ensure effective use by the trained answer reader. Historical access is therefore depth-selective without being equally effective at every depth.

4.2 Attenuation of Revisited Nodes

Among the 419 official test graphs, 154 contain a depth-one node revisited at step 3, and 132 contain a depth-two node revisited at step 4. These graphs are acyclic: a longer path can revisit a node without forming a cycle. Current-frontier nodes remain distinguishable from revisited nodes, whose cosine scores decrease on average from first arrival to later encounter (see Figure 6). Feedback thus preferentially represents the current frontier, although attenuation does not imply complete erasure of earlier information.

Furthermore, we also examine 59 answer-matched graph pairs with identical F_3 and $\mathcal{N}(F_3)$, but different visitation histories. With the recipient prompt, z_3, and other cache positions fixed, transplanting P_2 KV across all layers changes the revisited node's cosine score in z_4 by only -0.00105, whereas transplanting z_3 changes it by $+0.0539$; signs are aligned with donor history. Full results are in Appendix G. These interventions show that, in practice, feedback may carry update-relevant information beyond frontier membership.

Note that our construction uses exact FFN cleanup to retain only frontier information in feedback, yet training may not strictly enforce this operation. The additional context is therefore compatible

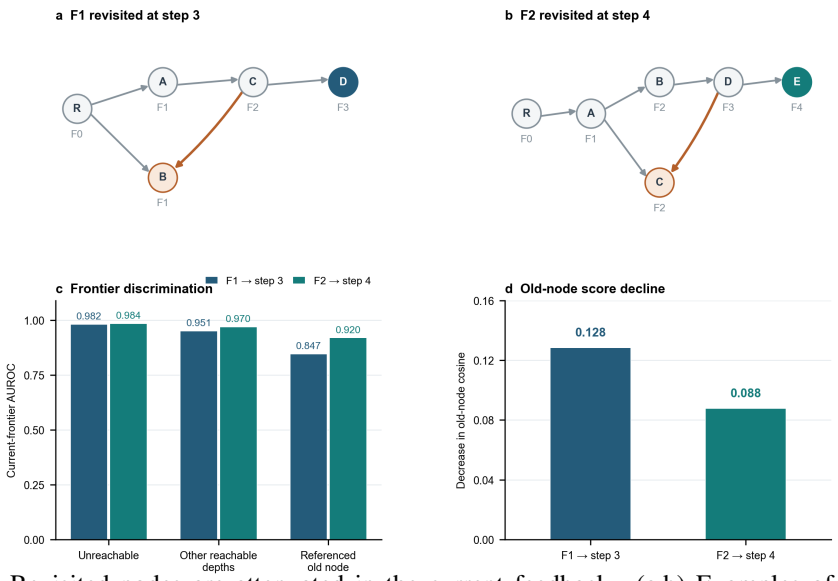

Figure 6: Revisited nodes are attenuated in the current feedback. (a,b) Examples of old nodes encountered again during expansion at steps 3 and 4. (c) Cosine-score AUROC for current-frontier nodes against unreachable nodes, nodes at other reachable depths, and revisited nodes. The two settings use 154 and 132 graphs, respectively. (d) Decrease in the revisited node's cosine score from its first arrival to its later encounter; positive values indicate attenuation.

with incomplete cleanup. Together with the historical readout results, the evidence suggests frontier-selective feedback with retained context alongside useful cached history.

5 Other Related Work

Latent reasoning and iterative computation. Related approaches compress reasoning traces into latent representations (Cheng & Van Durme, 2024) or train continuous token mixtures for parallel exploration (Gozeten et al., 2026). Computational analyses study the additional power supplied by CoT steps (Feng et al., 2023; Merrill & Sabharwal, 2024) and repeated Transformer blocks (Giannou et al., 2023; Saunshi et al., 2025). Graph reachability and shortest-path tasks provide controlled settings for these questions (Merrill & Sabharwal, 2025; Cohen et al., 2025). Our focus is how reasoning distributes information between recurrent feedback and persistent attention history.

Mechanistic evidence. Probing and interventions characterize latent representations, but decodable structure need not reflect the computation used by the model (Dilgren & Wiegreffe, 2026; Rizvi-Martel et al., 2026). Cache-level interventions investigate the contribution of stored values to answer prediction (Ding et al., 2026). We connect these perspectives by examining feedback and historical access separately during latent generation and readout, then using a constructive graph model to derive testable predictions about their respective roles.

We defer a complete discussion of related work to Appendix A.

6 Concluding Remarks

We study how continuous reasoning uses recurrent feedback and KV history. Our construction establishes that exact graph reasoning does not require cumulative storage in every feedback vector. In the trained model, feedback emphasizes the current frontier while retaining additional update-relevant context, and earlier cached states support depth-selective readout. These findings motivate studying feedback and history jointly.

Our analysis establishes computational sufficiency but does not characterize how this organization emerges through learning. Future work could extend intermediate and answer supervision, as in SIM-CoT (Wei et al., 2025), to shape feedback and cached representations for their respective uses. Learned history-selection policies could further improve access to relevant intermediate states during reasoning and readout.

AI USE STATEMENT

We used generative AI tools (e.g., GPT-6) to polish the language, correct grammatical errors, and improve the readability of the manuscript. These tools also assisted in refining our initial two-layer Transformer construction, particularly in selecting thresholds and parameters for FFN, and in implementing some of the experiments. The authors have reviewed all AI-assisted content and take full responsibility for the final work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work investigates the mechanisms of continuous chain-of-thought reasoning through theoretical analysis and controlled experiments on reasoning benchmarks. It does not involve human participants or the collection of personal or sensitive data. We do not identify ethical concerns specific to this study beyond those generally associated with research on language models.

REPRODUCIBILITY STATEMENT

Experimental protocols, native-task baselines, cross-model evaluations, and counterfactual controls are documented in Appendices B.1 , D and G. Checkpoint provenance and training configurations are given in Appendix H. Code, configuration files, and the simplified ProsQA train, validation, and test splits used for our graph-model training and evaluation will be available at `https://anonymous.4open.science/r/code-available-AD80`.

REFERENCES

Darpan Aswal, Thomas Palmeira Ferraz, Yongxin Zhou, and Maxime Peyrard. Observable patterns are not explanations: A causal-geometric analysis of latent reasoning models, 2026. URL https://arxiv.org/abs/2606.12689.

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

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.

Andrew Cohen, Andrey Gromov, Kaiyu Yang, and Yuandong Tian. Spectral journey: How transformers predict the shortest path. *arXiv preprint arXiv:2502.08794*, 2025. URL https://arxiv.org/abs/2502.08794.

Connor Dilgren and Sarah Wiegreffe. Are latent reasoning models easily interpretable? In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=PBwyWJFZJR.

Yi Ding, Lijun Huang, and Menglin Yang. SCIT: Testing causal cache carriers in latent chain-of-thought models. *arXiv preprint arXiv:2608.27265*, 2026. URL https://arxiv.org/abs/2608.27265.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers, 2026. URL https://arxiv.org/abs/2606.31779.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qHrADgAdYu.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*,

volume 38, Main Conference, pp. 41340–41391. Curran Associates, Inc., 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Paper-Conference.pdf.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 11398–11442. PMLR, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Alperen Gozeten, Muhammed Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 82881–82914, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/85ec26ef94c3acb4c195e905df1ff4f7-Paper-Conference.pdf.

Hongyu Gu, Chang Liu, and Jingwen Fu. How many thoughts can a vector hold? the capacity of reasoning by superposition, 2026. URL https://arxiv.org/abs/2609.13747.

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.

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*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3309b4112c9f04a993f2bbdd0274bba1-Abstract-Conference.html.

Zirui Li, Xuefeng Bai, Kehai Chen, Yizhi LI, Jian Yang, Chenghua Lin, and Min Zhang. Dynamics within latent chain-of-thought: An empirical study of causal structure. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kHB8m3ojGe.

Jia Liang and Liangming Pan. Do latent-cot models think step-by-step? a mechanistic study on sequential reasoning tasks, 2026. URL https://arxiv.org/abs/2602.00449.

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://proceedings.iclr.cc/paper_files/paper/2024/hash/1f59721c106ea80f613299039112f651-Abstract-Conference.html.

William Merrill and Ashish Sabharwal. A little depth goes a long way: the expressive power of log-depth transformers, 2025. URL https://openreview.net/forum?id=zDze7VtB5C.

Michael Rizvi-Martel, Guillaume Rabusseau, and Marius Mosbach. The illusion of superposition? a principled analysis of latent thinking in language models, 2026. URL https://arxiv.org/abs/2604.06374.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.17416.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16042.

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*, 2025. URL https://openreview.net/forum?id=1cD9iO5Isv.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Emergence of superposition: Unveiling the training dynamics of chain of continuous thought. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/658e8b85ff800877233d544b7d4f7069-Abstract-Conference.html.

A Related Work

Explicit and Latent Chain-of-Thought Reasoning Explicit CoT improves multi-step reasoning through intermediate textual traces (Wei et al., 2022), but increases generation cost. Continuous reasoning methods instead operate on dense representations: Coconut feeds hidden states back as subsequent inputs, while CODI, CCoT, and SIM-CoT explore latent reasoning through distillation, compression, and supervised latent objectives (Hao et al., 2025; Shen et al., 2025; Cheng & Van Durme, 2024; Wei et al., 2025). CoT2 explicitly trains continuous token mixtures to support parallel exploration (Gozeten et al., 2026). These methods motivate examining how reasoning information is organized within a latent trajectory.

Iterative Computation and Theoretical Analyses Theoretical analyses show that CoT expands the computational power of Transformers through additional sequential steps (Feng et al., 2023; Merrill & Sabharwal, 2024; Li et al., 2024). Looped Transformers likewise perform iterative computation through repeated layer applications (Giannou et al., 2023; Saunshi et al., 2025; Geiping et al., 2025). For continuous CoT, superposition-based analyses study graph reachability and the emergence of search representations (Zhu et al., 2025; 2026). We complement these capability and training analyses by separating recurrent state updates from persistent attention memory.

Mechanistic and Causal Studies of Latent Reasoning Mechanistic studies examine intermediate variables and dependencies across latent steps through probing and interventions (Liang & Pan, 2026; Li et al., 2026). Interpretability analyses caution that decodable patterns alone do not establish causal use (Dilgren & Wiegreffe, 2026; Aswal et al., 2026; Rizvi-Martel et al., 2026), while cache interventions investigate value states as causal carriers (Ding et al., 2026). Building on this work, we distinguish the roles of recurrent feedback and KV history during latent generation and answer readout.

Graph Reasoning and Frontier-Based Computation Graph reachability and shortest-path tasks provide controlled settings for studying Transformer reasoning (Zhu et al., 2025; Merrill & Sabharwal, 2025; Cohen et al., 2025). Breadth-first search distinguishes the current frontier, which determines expansion, from previously visited nodes, which suppress revisits. We use this distinction to construct a frontier–history model: feedback carries the current frontier, while cached states preserve earlier frontiers. This organization yields depth-selective predictions for history readout that we test in trained continuous-CoT models.

B Feedback and Persistent History: Cross-Model Causal Evidence

B.1 Evaluation Protocol and Native Reasoning Budgets

We evaluate GPT-2 Coconut, Llama-Coconut, SIM-CoT, CODI, and an explicit-CoT control on GSM8K, GSMHard, SVAMP, and MultiArith. GPT-2 Coconut uses three native reasoning steps with two latent positions per step, for a budget of six latent positions. Llama-Coconut uses six one-position steps, also for a budget of six. SIM-CoT uses five steps with two positions per step, for a budget of ten. CODI uses six recurrent latent slots without a separate multi-position step grouping. For explicit CoT, one generated «`expression=result`» block counts as one step, so its budget varies across examples.

All models use greedy decoding. Accuracy is exact match after normalizing the final generated number, and AnsMatch is the percentage of examples for which an intervention preserves the clean model's normalized answer. Readout interventions are applied only after the clean reasoning trajectory has been generated; question KV remains visible throughout. For explicit CoT, the textual rationale is fixed and masking applies only to the answer-boundary and answer-token queries; its reported N includes examples with at least one successfully parsed calculation block. We distinguish native-budget comparisons from the fixed-six-position sweep below. In the latter, all latent models are evaluated with six latent positions, while SIM-CoT's ten-position native baseline is reported separately. Each intervention is compared with the clean run from the same evaluation; paired deltas and AnsMatch, rather than small cross-table differences in clean accuracy, are the relevant quantities.

The experiments are organized around three implications. First, degradation under last-one or none readout tests whether the final cached position is sufficient for the model's native reader. Second, within-model restoration tests whether earlier cached states contribute independently of the retained final latent input. Third, phase-specific restrictions compare this persistent history with recurrent feedback during latent generation.

B.2 Readout Depends on Distributed Latent History

We first test whether answer-relevant information has been consolidated into the final cached latent position by the time latent reasoning terminates. The *last-one* condition is a direct sufficiency test: the generated trajectory is held fixed, but answer readout can access only its final latent position. The *none* condition removes latent-history access altogether. For explicit CoT, the analogous last-one intervention retains only the final calculation block. The table also reports *thinking prev-1*, which regenerates each latent trajectory while retaining recurrent feedback, question access, and only the immediately preceding latent KV; final readout is unrestricted.

				Thinking prev-1			Readout last-one			Readout none		
Model	Dataset	N	Clean Acc.	Acc.	ΔAcc	AnsMatch	Acc.	ΔAcc	AnsMatch	Acc.	ΔAcc	AnsMatch
GPT-2 Coconut	GSM8K	1319	34.42	31.46	-2.96	70.51	16.53	-17.89	35.03	5.69	-28.73	11.52
	GSMHard	1319	8.49	7.20	-1.29	59.51	4.09	-4.40	17.97	1.52	-6.97	6.14
	SVAMP	1000	36.10	36.50	0.40	80.00	29.40	-6.70	41.50	22.10	-14.00	27.20
	MultiArith	180	83.89	82.22	-1.67	91.11	32.22	-51.67	32.22	4.44	-79.44	7.22
Llama-Coconut	GSM8K	1319	36.01	35.71	-0.30	97.27	34.12	-1.90	75.21	28.89	-7.13	61.71
	GSMHard	1319	8.19	8.19	0.00	94.84	7.66	-0.53	62.70	6.67	-1.52	52.69
	SVAMP	1000	48.40	48.70	0.30	99.00	47.60	-0.80	90.10	45.10	-3.30	75.00
	MultiArith	180	86.67	86.67	0.00	100.00	81.67	-5.00	90.56	46.67	-40.00	53.89
SIM-CoT	GSM8K	1319	44.66	30.25	-14.40	50.27	30.33	-14.33	44.05	5.76	-38.89	9.40
	GSMHard	1319	9.33	6.60	-2.73	37.91	6.22	-3.11	16.22	0.76	-8.57	4.55
	SVAMP	1000	40.60	39.20	-1.40	67.00	33.80	-6.80	50.70	19.40	-21.20	23.20
	MultiArith	180	92.78	87.22	-5.56	91.11	67.22	-25.56	70.00	6.67	-86.11	9.44
CODI	GSM8K	1319	41.62	37.07	-4.55	75.13	21.15	-20.47	25.02	22.52	-19.11	24.79
	GSMHard	1319	9.33	8.64	-0.68	61.26	5.46	-3.87	10.39	5.16	-4.17	8.64
	SVAMP	1000	40.60	40.40	-0.20	89.20	34.10	-6.50	42.10	29.20	-11.40	32.90
	MultiArith	180	94.44	94.44	0.00	99.44	66.11	-28.33	67.22	27.22	-67.22	27.22
Explicit CoT	GSM8K	1309	42.25	–	–	–	42.25	0.00	99.47	2.83	-39.42	2.75
	GSMHard	1147	9.85	–	–	–	9.85	0.00	95.55	0.78	-9.07	1.31
	SVAMP	980	41.84	–	–	–	41.84	0.00	99.69	2.04	-39.80	1.53
	MultiArith	180	89.44	–	–	–	89.44	0.00	100.00	5.00	-84.44	5.00

Table 4: Native-budget phase and readout interventions. Acc., ΔAcc, and AnsMatch are percentages. Thinking prev-1 retains recurrent feedback, question KV, and the immediately preceding latent KV while regenerating the trajectory, followed by unrestricted readout. Last-one and none retain question KV and restrict only answer readout after clean latent generation. Thinking prev-1 is not defined for explicit CoT.

Table 4 shows that final-position sufficiency does not generally hold for the latent models. The strength of the effect varies substantially across architectures and datasets, but restricting readout to the last latent position changes many clean answers, and removing the entire latent cache changes still more. In contrast, retaining the final explicit calculation block leaves explicit-CoT accuracy essentially unchanged, indicating that the latent-model effect is not an inevitable consequence of hiding earlier reasoning positions. To distinguish dependence on a single critical position from dependence distributed across the trajectory, we additionally perform single-position and cumulative masking under a common six-position budget (Tables 5 and 6). Removing any one position is typically much less damaging than removing multiple positions together. The cumulative trends are not strictly monotonic, and the affected positions differ across models, so these interventions do not support a universal ordering of latent-position importance. Instead, they show that the trained reader can depend jointly on information distributed across multiple cached positions.

This picture is consistent with causal studies reporting non-local routing and stage-dependent roles across latent reasoning steps (Li et al., 2026), as well as mechanistic evidence that intermediate CODI states can be routed to a later readout and combined only near the end-of-thought boundary

(Liang & Pan, 2026). Our intervention isolates a different object: the accessibility of already-generated latent KV states to the native answer reader. Accordingly, the result establishes distributed readout dependence.

Model	Dataset	N	Native	Six clean	Mask z_1	Mask z_2	Mask z_3	Mask z_4	Mask z_5	Mask z_6
CODI	GSM8K	1319	41.62	41.62	36.01	42.38	40.11	41.70	36.69	41.24
	GSMHard	1319	9.33	9.33	7.81	9.40	9.17	9.48	8.87	9.40
	SVAMP	1000	40.60	40.60	40.90	40.70	41.00	40.40	41.00	40.40
	MultiArith	180	94.44	94.44	82.78	94.44	92.78	93.33	92.78	93.89
GPT-2 Coconut	GSM8K	1319	34.42	34.42	34.42	29.11	34.12	27.45	34.50	31.84
	GSMHard	1319	8.49	8.49	8.49	7.28	8.04	6.52	8.11	8.11
	SVAMP	1000	36.10	36.10	34.90	34.60	36.40	35.70	36.20	36.10
	MultiArith	180	83.89	83.89	83.33	58.89	75.56	80.56	83.89	83.89
Llama-Coconut	GSM8K	1319	36.01	36.01	36.16	35.78	35.94	35.78	35.94	35.86
	GSMHard	1319	8.19	8.19	8.19	8.26	8.19	8.11	8.11	8.11
	SVAMP	1000	48.40	48.40	48.30	48.40	48.40	48.40	48.30	48.20
	MultiArith	180	86.67	86.67	84.44	85.56	85.00	84.44	85.56	85.56
SIM-CoT	GSM8K	1319	44.66	38.21	38.59	37.00	37.53	35.56	35.63	34.95
	GSMHard	1319	9.33	8.64	8.64	8.42	8.26	8.34	7.96	8.11
	SVAMP	1000	40.60	40.70	39.70	39.80	40.50	39.90	40.10	40.10
	MultiArith	180	92.78	92.78	92.78	87.22	90.56	92.22	92.78	92.22

Table 5: Accuracy (%) when exactly one of six latent positions is masked during readout. Native is the clean accuracy at the model's native latent budget; Six clean is the unmasked six-position reference. Each Mask z_i condition removes only position i while retaining the other five positions and all question KV.

We next mask increasingly long suffixes of the same six-position trajectory. The conditions in Table 6 range from removing only z_6 to removing $z_1, \ldots, z_6$.

Model	Dataset	N	Six clean	Mask z_6	Mask $z_{5:6}$	Mask $z_{4:6}$	Mask $z_{3:6}$	Mask $z_{2:6}$	Mask $z_{1:6}$
CODI	GSM8K	1319	41.62	41.24	36.92	37.00	22.59	25.17	22.52
	GSMHard	1319	9.33	9.40	9.10	8.79	4.85	5.53	5.16
	SVAMP	1000	40.60	40.40	41.20	40.80	39.80	35.10	29.20
	MultiArith	180	94.44	93.89	92.22	91.67	66.67	47.22	27.22
GPT-2 Coconut	GSM8K	1319	34.42	31.84	30.48	20.32	18.27	8.34	5.69
	GSMHard	1319	8.49	8.11	7.28	5.00	4.55	2.20	1.52
	SVAMP	1000	36.10	36.10	35.70	35.80	35.40	32.00	22.10
	MultiArith	180	83.89	83.89	83.33	80.56	70.00	5.56	4.44
Llama-Coconut	GSM8K	1319	36.01	35.86	35.25	35.10	34.42	32.98	28.89
	GSMHard	1319	8.19	8.11	7.96	7.96	7.73	7.58	6.67
	SVAMP	1000	48.40	48.20	48.10	47.70	47.30	46.50	45.10
	MultiArith	180	86.67	85.56	84.44	80.00	75.56	64.44	46.67
SIM-CoT	GSM8K	1319	38.21	34.95	28.20	20.47	13.72	5.91	5.46
	GSMHard	1319	8.64	8.11	6.37	5.00	2.96	0.83	0.68
	SVAMP	1000	40.70	40.10	40.00	39.10	35.00	31.50	19.40
	MultiArith	180	92.78	92.22	92.78	90.00	47.78	7.78	5.56

Table 6: Accuracy (%) under reverse cumulative masking of the six latent positions. Mask $z_{i:6}$ removes positions i through 6 only during readout, after an unchanged six-position trajectory has been generated; question KV remains visible. Six clean is the reference for every masking condition in this table.

The last-one and none interventions directly reject final-cache sufficiency for the native reader: retaining only the final cached position changes many answers in the latent models, and removing the complete latent cache changes still more. The fixed-six sweep gives the complementary positional result. No single position is universally decisive, whereas removing several positions together often causes a much larger loss. The cumulative declines are not strictly monotonic, especially for CODI,

so the result supports distributed readout dependence rather than a universal ordering of position importance.

B.3 Earlier Cached History Has an Independent Causal Contribution

The preceding masking results establish dependence on latent history but do not by themselves distinguish information stored in the earlier cache from information already summarized in the final recurrent input. We therefore use a within-model corruption-and-restoration intervention related to activation patching (Zhang & Nanda, 2024). For each target example, we replace its earlier latent inputs with those from a donor example while retaining the target's final latent input. Starting independently from this corrupted run, we then restore selected KV entries from the target's clean trajectory. No tensors are transferred across model architectures.

Model	Dataset	N	Clean	Corrupted	Final KV	Earlier KV	Earlier K	Earlier V
GPT-2 Coconut	GSM8K	1319	34.42	5.23	8.19	33.81	6.29	24.87
	GSMHard	1319	8.49	1.14	1.74	8.19	1.44	5.23
	SVAMP	1000	36.10	12.50	13.40	36.00	17.60	32.70
	MultiArith	180	83.89	3.89	9.44	82.78	5.00	68.89
Llama-Coconut	GSM8K	1319	35.94	29.95	29.57	35.86	26.91	36.16
	GSMHard	1319	8.19	7.43	7.51	8.19	6.90	8.19
	SVAMP	1000	48.40	45.80	45.90	48.40	45.00	48.30
	MultiArith	180	86.67	53.89	53.33	86.11	51.11	86.11
SIM-CoT	GSM8K	1319	44.66	4.85	9.10	44.43	5.16	38.82
	GSMHard	1319	9.33	1.14	2.05	9.17	1.67	8.11
	SVAMP	1000	40.60	8.20	10.70	40.50	10.00	35.70
	MultiArith	180	92.78	6.11	7.78	92.78	4.44	91.11
CODI	GSM8K	1319	41.93	3.94	4.09	41.85	5.00	35.18
	GSMHard	1319	9.33	0.91	0.91	9.33	1.06	8.79
	SVAMP	1000	40.50	20.20	20.50	40.40	32.60	37.00
	MultiArith	180	94.44	4.44	5.00	94.44	5.00	88.33

Table 7: Accuracy (%) under within-model latent-history corruption and KV restoration. Corrupted combines a same-model donor's earlier latent inputs with the target's final latent input. Each restoration condition starts independently from Corrupted: Final KV restores the target's final-position keys and values; Earlier KV restores both keys and values at all earlier latent positions; Earlier K and Earlier V restore only the indicated tensor at those earlier positions.

The relevant comparison in Table 7 is recovery from the corrupted baseline. Whenever corruption produces a substantial loss, restoring the earlier KV entries recovers most or all of the clean performance, whereas restoring only the final-position KV produces little recovery. Moreover, restoring earlier values alone typically recovers substantially more than restoring earlier keys alone. Because every corrupted condition retains the target's final recurrent input, the restored behavior cannot be explained by information available only through that final input or through the final cached position.

The stronger contribution of earlier values is closely aligned with SCIT, which identifies value-cache trajectories as the principal causal carrier in several arithmetic latent-CoT settings (Ding et al., 2026). The two analyses ask complementary questions: SCIT localizes which cache component can carry a counterfactual computation, whereas our intervention asks whether earlier cached history contributes information that remains causally useful even when the target's final recurrent input is preserved. The result therefore supports a persistent-history component distinct from recurrent state propagation.

B.4 Feedback and Persistent History Play Complementary Roles

Readout dependence on persistent history does not imply that the same history is required while the latent trajectory itself is being generated. We test this distinction with *thinking prev-1*: each

latent position retains recurrent feedback, question access, self-attention, and only the immediately preceding latent KV. The resulting trajectory is then read out with unrestricted access. By contrast, the last-one and none conditions leave latent generation unchanged and restrict history only during answer readout.

Table 4 shows a broad but architecture-dependent phase asymmetry. In 15 of the 16 latent-model–dataset settings, thinking prev-1 loses less accuracy than readout last-one; in all 16, it loses less than readout none. SIM-CoT on GSM8K is the exception to the first comparison: thinking prev-1 and last-one cause nearly identical drops of 14.40 and 14.33 percentage points. Thus, recurrent feedback plus local history often advances the latent computation more robustly than the final reader tolerates loss of persistent history, but full-history dependence during thinking remains model- and task-dependent.

Recurrent feedback primarily propagates computation forward through latent positions, while persistent KV history provides the final reader with access to information that need not have been consolidated into the last cached position. This phase-specific organization is compatible with prior evidence that different latent steps can serve distinct causal functions rather than behaving as homogeneous additional depth (Li et al., 2026).

B.5 The Role of Latent History Is Task Dependent

The arithmetic experiments above should not be read as a universal claim that latent trajectories are necessary on every task. We therefore evaluate the same distinction on PrOntoQA (Saparov & He, 2023) and ProsQA (Hao et al., 2025) using public GPT-2 Coconut checkpoints (Dilgren & Wiegreffe, 2026). In addition to the phase-specific history interventions, we include zero-latent controls that either preserve the answer position with masked placeholders or remove the latent slots entirely.

Condition	PrOntoQA $n = 800$	ProsQA $n = 500$
Baseline (6 latents)	99.000	98.000
Thinking prev-1	99.000	98.000
Readout only-last1	99.125	98.000
Readout none	99.125	98.000
Zero-latent: preserve positions, mask placeholder keys	99.125	98.000
Zero-latent: remove slots (compact)	99.125	98.000

Table 8: Phase interventions and zero-latent controls on PrOntoQA and ProsQA. Entries are accuracies in percent. Both zero-latent variants skip recurrent feedback; only the position-preserving variant controls the absolute answer position. Readout none masks all latent KV while retaining question KV.

As shown in Table 8, eliminating the recurrent latent rollout does not reduce accuracy on either logical-reasoning task. The same accuracy is also retained when all latent KV is hidden during readout. These controls indicate that, for these checkpoints and tasks, the additional latent trajectory is not required to sustain the observed performance; question processing and answer-side computation suffice under this evaluation.

This result is consistent with the early-stopping findings of Dilgren & Wiegreffe (2026), who similarly observe that latent reasoning tokens are often unnecessary for stable predictions on PrOntoQA and ProsQA. Our controls sharpen that observation by separating skipped recurrent computation from positional effects and from latent-history access during readout. The conclusion is therefore task- and checkpoint-specific: preserved zero-latent accuracy shows that the additional rollout is unnecessary under this evaluation.

C Detailed Construction of the Two-Block Transformer

This appendix gives the coordinate-level realization of the 2-block transformer description. It specifies the attention projections, finite-softmax separation margins, shared FFN construction, and resid-

ual cleanup procedure. All parameters below depend only on the vertex bound N, edge bound $M_{\max}$, and maximum causal context length $P_{\max}$; they do not depend on a particular graph or reasoning step.

C.1 State layout and notation

We use a common residual-state layout for both transformer blocks. At every sequence position, the hidden state is partitioned into the fixed coordinate order

$$h = [X; \pi; S; T; C; U; \mathrm{role}; W; \mathrm{pad}], \tag{5}$$

where

$$\begin{aligned} X \in \mathbb{R}^{N+6}, \qquad \pi \in \mathbb{R}^2, \quad S, T, C \in \mathbb{R}^N, \\ U \in \mathbb{R}^{N+1}, \quad \mathrm{role} \in \mathbb{R}^6, \qquad W \in \mathbb{R}^{5N+1}. \end{aligned} \tag{6}$$

The remaining coordinates form an unused padding block. The registers have the following roles:

register	meaning
X	input token or recurrent-feedback content
π	positional coordinates
S	source node of an edge record
T	target node of an edge record
C	the two answer candidates
U	current frontier, together with one anchor coordinate
role	position type
W	temporary outputs of the attention heads.

The content register has the internal order

$$X = [X_{\mathcal{V}}; X_\star; X_s; X_e; X_Q; X_R; X_A], \tag{7}$$

where $X_{\mathcal{V}} \in \mathbb{R}^N$ contains the node coordinates, $X_\star$ is the anchor coordinate, and X_s, X_e, X_Q, X_R, X_A are the coordinates associated with $\langle s\rangle, \langle e\rangle, \langle Q\rangle, \langle R\rangle, \langle A\rangle$, respectively. For a node v, let e_v denote its standard basis vector in the N node coordinates, and let $e_\star$ denote the anchor basis vector.

The role register has the fixed order

$$\mathrm{role} = [E; R; L; A; S_0; D_0], \tag{8}$$

corresponding respectively to an edge record, the candidate record, a root or recurrent frontier position, the answer position, the initial sink position, and an otherwise inert position. We write

$$e_E,\ e_R,\ e_L,\ e_A,\ e_{S_0},\ e_{D_0}$$

for the corresponding standard basis vectors in the six-dimensional role register. Input-token flags such as X_R and output-role coordinates such as R are distinct coordinates.

Finally, the work register is partitioned as

$$W = [c^{(1)}; c^{(2)}; m; a; b; g], \tag{9}$$

with

$$c^{(1)}, c^{(2)}, a, b, g \in \mathbb{R}^N, \qquad m \in \mathbb{R}.$$

The first three components are used by Block 1: $c^{(1)}$ and $c^{(2)}$ store the node-copy outputs from one and two positions back, and m stores the marker-head output. The last three components are used by Block 2: a, b, and g store the expansion, visited-history, and candidate-head outputs, respectively. Each block clears the full work register before producing its normalized output.

All coordinate blocks in equation 5 are disjoint. Consequently, linear projections may independently read from or write to any specified register, and the normalization factors used below can be computed directly from the squared norms of the nonzero blocks.

C.2 Reusable ReLU primitives and residual cleanup

We first record two elementary ReLU operations used in the FFN constructions below.

Threshold ramp. Write $\sigma(x) = \max\{x, 0\}$, applied coordinatewise to vector arguments. For any $\tau_- < \tau_+$, define

$$H_{\tau_-,\tau_+}(x) = \frac{\sigma(x - \tau_-) - \sigma(x - \tau_+)}{\tau_+ - \tau_-}. \tag{10}$$

Then

$$H_{\tau_-,\tau_+}(x) = \begin{cases} 0, & x \le \tau_-, \\ \dfrac{x - \tau_-}{\tau_+ - \tau_-}, & \tau_- < x < \tau_+, \\ 1, & x \ge \tau_+. \end{cases}$$

Thus the ramp takes values in $[0, 1]$ and is exactly binary outside its transition interval. Whenever legal inputs are separated from this interval by a finite margin, it therefore recovers the corresponding indicator exactly. We write $H = H_{1/4,3/4}$ for the fixed ramp used in Block 1.

Conditional gate. For $x \in [0, 1]$ and an affine control variable t, define

$$G(x, t) = \sigma\big(x + 4(t - 1/2)\big) - \sigma\big(4(t - 1/2)\big). \tag{11}$$

For the regimes used below,

$$G(x, t) = \begin{cases} x, & t \ge 1, \\ 0, & t \le 0. \end{cases} \tag{12}$$

Thus G conditionally preserves an existing coordinate rather than thresholding it. Both H and G are linear combinations of two ReLUs of affine functions of the same input, and hence can be evaluated within a single ReLU hidden layer.

Residual cleanup. The FFNs below must both compute new coordinates and erase temporary information inherited through the residual connection. The following construction allows these operations to be performed simultaneously.

Let $u \in \mathbb{R}^d$ be the state after an attention residual, and suppose the desired cleaned state has the form

$$\mathcal{C}(u) = Du + B\sigma(Au + a) + b, \tag{13}$$

where D, A, B are fixed matrices and a, b are fixed biases. Thus $\mathcal{C}$ may preserve, erase, or linearly transform existing coordinates, while also writing coordinates obtained from affine ReLU operations.

Since a transformer block returns

$$u + \mathrm{FFN}(u),$$

it suffices to implement

$$\mathrm{FFN}(u) = \mathcal{C}(u) - u.$$

Using the coordinatewise identity

$$\sigma(u) - \sigma(-u) = u, \tag{14}$$

the required linear correction can be implemented within the same hidden layer. More explicitly, if the nonlinear part of equation 13 uses w ReLU units, take

$$W_{\mathrm{in}} = \begin{bmatrix} A \\ I_d \\ -I_d \end{bmatrix}, \qquad b_{\mathrm{in}} = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}, \tag{15}$$

and

$$W_{\mathrm{out}} = \begin{bmatrix} B & D - I_d & -(D - I_d) \end{bmatrix}. \tag{16}$$

With output bias b,

$$\begin{aligned} \mathrm{FFN}(u) &= B\sigma(Au + a) + b + (D - I_d)\big[\sigma(u) - \sigma(-u)\big] \\ &= B\sigma(Au + a) + b + (D - I_d)u \\ &= \mathcal{C}(u) - u. \end{aligned} \tag{17}$$

Therefore

$$u + \mathrm{FFN}(u) = \mathcal{C}(u) \tag{18}$$

exactly.

Hence one affine ReLU hidden layer can simultaneously perform the thresholding operations used below, erase arbitrary residual coordinates, and write the desired canonical state. The w task-specific ReLU units and the $2d$ residual-correction units operate in parallel, so this cleanup requires no additional FFN depth.

C.3 Block 1: Record construction

Use the fixed coordinate order

$$h = [X; \pi; S; T; C; U; role; W; \mathrm{pad}], \tag{19}$$

where $X \in \mathbb{R}^{N+6}$ contains lexical coordinates, $\pi \in \mathbb{R}^{d_{\mathrm{PE}}}$ contains positional coordinates, $S, T, C \in \mathbb{R}^N$ contain edge-source, edge-target, and candidate records, $U \in \mathbb{R}^{N+1}$ contains the frontier and anchor coordinates, $role \in \mathbb{R}^6$ contains role coordinates, and $W \in \mathbb{R}^{5N+1}$ is scratch space. Let $e_E, e_R, e_L, e_A, e_{S_0}, e_{D_0}$ be the corresponding role basis vectors.

Fix a position and omit its subscript. After the Block 1 attention sublayer and residual addition, the original content and positional coordinates X, π remain in the residual stream, while the three attention heads write

$$\mathrm{work}^{(1)} = \begin{bmatrix} c^{(1)} \\ c^{(2)} \\ m \\ 0_{3N} \end{bmatrix}.$$

Here $c^{(1)}, c^{(2)} \in \mathbb{R}^N$ are the node-copy outputs from one and two positions back, respectively, and m is the output of the uniform marker head. Let

$$\bar{X} = (X_1, \ldots, X_N, X_\star)^\top$$

denote the node-and-anchor content, and let X_e, X_R, X_A, X_s denote the input flags for $\langle e \rangle, \langle R \rangle, \langle A \rangle, \langle s \rangle$.

Phase control. For the marker head, $m = 0$ before $\langle R \rangle$ is visible and $m = 1/i$ thereafter. Define the affine control

$$t = P_{\max} m - P_{\max}(X_R + X_A). \tag{20}$$

At every root or recurrent frontier position,

$$t = \frac{P_{\max}}{i} \geq 1.$$

At $\langle R \rangle$ and $\langle A \rangle$, the subtraction term gives $t \leq 0$, while before $\langle R \rangle$ is visible we have $m = 0$ and hence $t = 0$. Therefore the threshold ramp $H(t)$ is exactly one at frontier positions and zero elsewhere, while the conditional gate $G(\bar{X}, t)$ preserves the node-and-anchor content exactly at frontier positions and clears it at all other positions.

Record and role writes. Choose the copy-head scale so that, at every edge or candidate delimiter, the requested predecessor receives attention weight at least $7/8$. This is possible by the positive score gap in equation 4. Consequently, if the requested node is v, its copied coordinate is at least $7/8$, whereas every other coordinate is at most $1/8$. The fixed ramp $H = H_{1/4,3/4}$ therefore recovers the corresponding one-hot node vector exactly.

Writing $\mathbf{1}_N$ for the all-ones vector, define the four record blocks by

$$\begin{bmatrix} S \\ T \\ C \\ U \end{bmatrix} = \begin{bmatrix} H\big(c^{(2)} + 2(X_e - 1)\mathbf{1}_N\big) \\ H\big(c^{(1)} + 2(X_e - 1)\mathbf{1}_N\big) \\ H\big(c^{(2)} + 2(X_R - 1)\mathbf{1}_N\big) + H\big(c^{(1)} + 2(X_R - 1)\mathbf{1}_N\big) \\ G(\bar{X}, t) \end{bmatrix}. \tag{21}$$

The additive shifts suppress each copy away from its designated delimiter: because every coordinate of $c^{(1)}$ and $c^{(2)}$ lies in $[0, 1]$, the argument is at most -1 whenever the corresponding flag is zero.

Thus, at an edge delimiter, S and T recover the source and target nodes; at $\langle R \rangle$, C recovers the two candidate nodes; and at a root or recurrent frontier position, U copies the current node-and-anchor state.

The role coordinates are

$$\text{role} = \begin{bmatrix} E \\ R \\ L \\ A \\ S_0 \\ D_0 \end{bmatrix} = \begin{bmatrix} X_e \\ X_R \\ H(t) \\ X_A \\ X_s \\ 1 - X_e - X_R - H(t) - X_A - X_s \end{bmatrix}. \tag{22}$$

For every legal input position, exactly one of these six coordinates equals one and the others equal zero. Hence the role block is one-hot before normalization.

Residual cleanup and normalization. Let $\mathcal{C}_1$ denote the complete Block 1 cleanup map. It writes the record and role coordinates above and erases the original content, positional, work, and padding coordinates. In the fixed state order,

$$\mathcal{C}_1(u_i^{(1)}) = \begin{bmatrix} 0 \\ 0 \\ \hline S \\ T \\ C \\ U \\ \hline \text{role} \\ 0 \\ 0 \end{bmatrix}. \tag{23}$$

Every coordinate of $\mathcal{C}_1$ is either affine in $u_i^{(1)}$ or a linear combination of ReLUs of affine functions of $u_i^{(1)}$. Therefore $\mathcal{C}_1$ has the form of equation 13. By the residual-cleanup construction, a single one-hidden-layer ReLU FFN satisfies

$$u_i^{(1)} + \text{FFN}_1(u_i^{(1)}) = \mathcal{C}_1(u_i^{(1)}) \tag{24}$$

exactly.

Since the role block is one-hot, $\mathcal{C}_1(u_i^{(1)}) \neq 0$ at every legal position, so the subsequent Euclidean normalization is always well defined. The Block 1 output is therefore

$$h_i^{(1)} = \text{Norm}_2\left(\mathcal{C}_1(u_i^{(1)})\right). \tag{25}$$

In particular, the three canonical states used by Block 2 are

$$h_{\text{edge}(m)}^{(1)} = \frac{1}{\sqrt{3}}[0; 0; e_{s_m}; e_{t_m}; 0; 0; e_E; 0; 0], \tag{26}$$

$$h_R^{(1)} = \frac{1}{\sqrt{3}}[0; 0; 0; 0; e_{c_1} + e_{c_2}; 0; e_R; 0; 0], \tag{27}$$

$$h_{i_c}^{(1)} = \frac{1}{\sqrt{2}}[0; 0; 0; 0; 0; z_c; e_L; 0; 0]. \tag{28}$$

The normalization factors follow from the disjoint supports of the nonzero blocks:

$$\|e_{s_m}\|^2 + \|e_{t_m}\|^2 + \|e_E\|^2 = 3, \qquad \|e_{c_1} + e_{c_2}\|^2 + \|e_R\|^2 = 3,$$

and

$$\|z_c\|^2 + \|e_L\|^2 = 2.$$

Thus Block 1 converts the serialized prompt and recurrent input into the clean canonical edge, candidate, and frontier records consumed by Block 2.

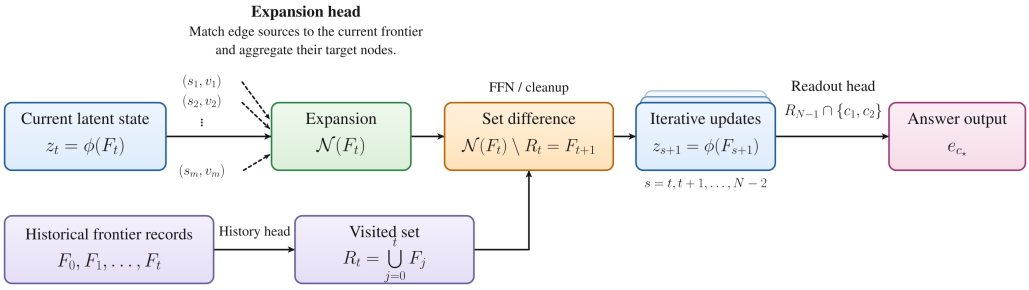

Figure 7: Frontier expansion, removal of visited nodes, and answer readout in the model.

C.4 Block 2: Frontier update and answer readout

Block 2 receives the canonical records produced by Block 1. Its three attention heads extract complementary information: the expansion head collects outgoing neighbors of the current frontier, the history head recovers all previously visited nodes, and the candidate head retrieves the two answer candidates. A shared ReLU FFN then combines these quantities to produce either the next frontier or the final answer.

Define

$$P_u = [I_N \quad 0_{N\times 1}], \tag{29}$$

which removes the anchor coordinate from the frontier register U.

C.4.1 Expansion head

The expansion head uses an N-dimensional node channel together with one scalar sink channel. In the state order of equation 5, its projections are

$$Q_{\text{exp}} = \sqrt{2}\alpha \left[\begin{array}{cc|cccc|ccc} 0_{N\times(N+6)} & 0_{N\times d_{\text{PE}}} & 0 & 0 & 0 & P_u & 0 & 0 & 0 \\ 0_{1\times(N+6)} & 0_{1\times d_{\text{PE}}} & 0 & 0 & 0 & 0 & e_L^\top & 0 & 0 \end{array}\right], \tag{30}$$

$$K_{\text{exp}} = \left[\begin{array}{cc|cccc|ccc} 0_{N\times(N+6)} & 0_{N\times d_{\text{PE}}} & \sqrt{3}I_N & 0 & 0 & 0 & 0 & 0 & 0 \\ 0_{1\times(N+6)} & 0_{1\times d_{\text{PE}}} & 0 & 0 & 0 & 0 & \kappa e_{S_0}^\top & 0 & 0 \end{array}\right], \tag{31}$$

and

$$V_{\text{exp}} = \left[\begin{array}{cc|cccc|ccc} 0_{N\times(N+6)} & 0_{N\times d_{\text{PE}}} & 0 & \sqrt{3}I_N & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{32}$$

Equivalently,

$$\begin{aligned} q_i^{\text{exp}} &= \sqrt{2}\alpha\big[P_u U_i;\, e_L^\top \text{role}_i\big], \\ k_j^{\text{exp}} &= \big[\sqrt{3}S_j;\, \kappa e_{S_0}^\top \text{role}_j\big], \\ v_j^{\text{exp}} &= \sqrt{3}T_j. \end{aligned} \tag{33}$$

At a frontier position i_c, the normalization factors in the canonical states of Block 1 are canceled by the fixed $\sqrt{2}$ and $\sqrt{3}$ factors, so the attention score becomes

$$(q_{i_c}^{\text{exp}})^\top k_j^{\text{exp}} = \begin{cases} \alpha(P_u z_c)^\top e_{s_m}, & j \text{ is the edge record } (s_m, t_m), \\ \alpha\kappa, & j = \langle s \rangle, \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

If D_c positions are causally visible at i_c, the node-channel output is

$$a_c = \sum_{m=1}^{M} \frac{\exp\big(\alpha(P_u z_c)^\top e_{s_m}\big)}{Z_c} e_{t_m}, \tag{35}$$

where

$$Z_c = \sum_{m=1}^{M} \exp\big(\alpha(P_u z_c)^\top e_{s_m}\big) + \exp(\alpha\kappa) + (D_c - M - 1). \tag{36}$$

Let

$$\delta = \frac{1}{\sqrt{N+1}}$$

and choose $\kappa = 1$. Every positive node coordinate in a nonempty frontier has magnitude at least δ. Moreover,

$$Z_c \le (M_{\max}+1)e^{\alpha} + P_{\max}, \qquad Z_c \ge e^{\alpha}.$$

Hence define

$$\mu_+ = \frac{e^{\alpha\delta}}{(M_{\max}+1)e^{\alpha} + P_{\max}}, \qquad \mu_- = M_{\max}e^{-\alpha}. \tag{37}$$

If $v \in \mathcal{N}(F_c)$, at least one matching edge contributes to coordinate v with score at least $\alpha\delta$, so

$$(a_c)_v \ge \mu_+.$$

If $v \notin \mathcal{N}(F_c)$, every edge ending at v has nonmatching source and hence score zero, giving

$$(a_c)_v \le \mu_-.$$

Since $\mu_+/\mu_- \to \infty$ as $\alpha \to \infty$, choose a finite α such that $\mu_+ > \mu_-$, and fix thresholds satisfying

$$\mu_- < \tau_- < \tau_+ < \mu_+. \tag{38}$$

Thus the expansion head separates true and false outgoing neighbors by a strict finite-softmax margin.

C.4.2 HISTORY HEAD

The history head uses uniform causal attention:

$$q_i^{\text{hist}} = 0, \qquad k_j^{\text{hist}} = 0, \qquad v_j^{\text{hist}} = \sqrt{2}P_u U_j. \tag{39}$$

Therefore all D_c visible positions receive weight $1/D_c$.

By equation 28, a frontier position containing z_s contributes value $P_u z_s$, while every nonfrontier position has zero U value. Consequently,

$$b_c = \frac{1}{D_c}\sum_{s=0}^{c} P_u z_s, \qquad \operatorname{supp}(b_c) = R_c. \tag{40}$$

All terms are nonnegative, so no cancellation is possible. Since $D_c \le P_{\max}$ and every positive frontier coordinate is at least δ, every visited node satisfies

$$(b_c)_v \ge \beta, \qquad \beta := \frac{1}{P_{\max}\sqrt{N+1}}, \tag{41}$$

whereas every unvisited node has $(b_c)_v = 0$.

C.4.3 CANDIDATE HEAD

The candidate head retrieves the candidate pair for use at the answer position. Its scalar query reads the answer role, its key reads the candidate-record role, and its value reads the candidate register. Define

$$\begin{aligned} q_i^{\text{cand}} &= \alpha_C\, e_A^{\top}\text{role}_i, \\ k_j^{\text{cand}} &= \sqrt{3}\, e_R^{\top}\text{role}_j, \\ v_j^{\text{cand}} &= \sqrt{3}\, C_j. \end{aligned} \tag{42}$$

By equation 27, the candidate-record position has

$$e_R^{\top}\text{role}_R = \frac{1}{\sqrt{3}}, \qquad C_R = \frac{1}{\sqrt{3}}(e_{c_1} + e_{c_2}),$$

so the fixed $\sqrt{3}$ factors undo the Block 1 normalization:

$$k_R^{\text{cand}} = 1, \qquad v_R^{\text{cand}} = e_{c_1} + e_{c_2}. \tag{43}$$

Every other position has zero candidate value.

At the answer position, the role coordinate gives $q_A^{\text{cand}} = \alpha_C$. Hence

$$(q_A^{\text{cand}})^\top k_j^{\text{cand}} = \begin{cases} \alpha_C, & j = R, \\ 0, & j \neq R. \end{cases} \tag{44}$$

If D_A positions are visible at the answer position, the attention output is therefore

$$g_A = \gamma_A(e_{c_1} + e_{c_2}), \qquad \gamma_A = \frac{e^{\alpha_C}}{e^{\alpha_C} + D_A - 1}. \tag{45}$$

The remaining softmax mass contributes no node content because all noncandidate positions have zero value in this head.

Choose α_C so that

$$e^{\alpha_C} \geq \frac{12 P_{\max}}{\beta}. \tag{46}$$

Since $D_A \leq P_{\max}$,

$$1 - \gamma_A = \frac{D_A - 1}{e^{\alpha_C} + D_A - 1} \leq \frac{P_{\max}}{e^{\alpha_C}} \leq \frac{\beta}{12}. \tag{47}$$

Thus each candidate coordinate in g_A is at least $1 - \beta/12$, which is sufficient for the answer-selection margin used by the shared FFN below.

At nonanswer positions the query is zero, so this head need not vanish identically; however, its output is irrelevant there because the answer branch of the shared FFN is disabled whenever the answer-role indicator χ is zero.

C.4.4 Shared FFN and residual cleanup

After the attention sublayer and residual addition, let $a, b, g \in \mathbb{R}^N$ denote the expansion, history, and candidate-head outputs in the work coordinates. Since Block 2 attention writes only to the work coordinates, the role coordinates are unchanged by the attention residual. Define the decoded indicators

$$\ell := \sqrt{2}\, e_L^\top \text{role}, \qquad \chi := e_A^\top \text{role}.$$

By the normalized canonical states of Block 1, these satisfy

$$(\ell, \chi) \in \{(1, 0), (0, 1), (0, 0)\}.$$

Frontier branch. At a frontier position, the desired next frontier is the set of outgoing neighbors that have not previously been visited. Define

$$f = H_{\tau_-, \tau_+}\left(a - \frac{2}{\beta} b - 2(1 - \ell)\mathbf{1}_N\right), \tag{48}$$

with H_{τ_-, τ_+} applied coordinatewise.

First consider a frontier position, so $\ell = 1$. If $v \in R_c$, then $b_v \geq \beta$. Since every attention coordinate satisfies $a_v \leq 1$,

$$a_v - \frac{2}{\beta} b_v \leq 1 - 2 = -1 < \tau_-,$$

so every previously visited node is removed.

If $v \notin R_c$, then $b_v = 0$. The expansion margin gives

$$\begin{cases} (a_c)_v \geq \mu_+, & v \in \mathcal{N}(F_c), \\ (a_c)_v \leq \mu_-, & v \notin \mathcal{N}(F_c). \end{cases} \tag{49}$$

Together with $\mu_- < \tau_- < \tau_+ < \mu_+$, this yields

$$f_v = \mathbf{1}\{v \in \mathcal{N}(F_c) \setminus R_c\} = \mathbf{1}\{v \in F_{c+1}\}. \tag{50}$$

Thus f is exactly the indicator vector of the next BFS frontier.

At every nonfrontier position, $\ell = 0$. Since $a_v \leq 1$ and $b_v \geq 0$, the additional -2 shift places every coordinate below τ_-, so $f = 0$ there.

Answer branch. At the answer position, the FFN must select the unique candidate that lies in the visited set. Define

$$y^{\mathrm{ans}} = H_{\beta/3,\,2\beta/3}\left(b + 3g - [3 + 5(1-\chi)]\mathbf{1}_N\right). \tag{51}$$

At the answer position, $\chi = 1$. Let $c_\star$ denote the reachable candidate. Since $c_\star \in R$, the history margin gives $(b_A)_{c_\star} \geq \beta$, while equation 47 gives $(g_A)_{c_\star} \geq 1 - \beta/12$. Therefore

$$\begin{aligned}(b_A)_{c_\star} + 3(g_A)_{c_\star} - 3 &\geq \beta + 3\left(1 - \frac{\beta}{12}\right) - 3 \\ &= \frac{3\beta}{4} > \frac{2\beta}{3}.\end{aligned} \tag{52}$$

Hence the reachable candidate coordinate is mapped to one.

For the unreachable candidate, the history coordinate is zero, and

$$3(g_A)_v - 3 = -3(1 - \gamma_A) \leq 0 < \frac{\beta}{3}.$$

For every noncandidate node, $g_v = 0$, while $b_v \leq 1$, so its argument is at most -2. Thus every other coordinate is below the lower threshold and

$$y^{\mathrm{ans}} = e_{c_\star}. \tag{53}$$

When $\chi = 0$, the additional -5 shift suppresses the entire answer branch, so $y^{\mathrm{ans}} = 0$ away from the answer position.

Residual cleanup and normalization. The two branches above produce exact node indicators but do not yet form the final Block 2 hidden state. We now use the residual-cleanup construction of Section C.2 to erase all auxiliary coordinates and write the desired canonical output.

Let

$$\widehat{X} = \begin{bmatrix} f + y^{\mathrm{ans}} \\ \ell \\ 0_5 \end{bmatrix} \in \mathbb{R}^{N+6}, \tag{54}$$

where the first N coordinates are the node coordinates and the next coordinate is the anchor coordinate. Thus the anchor is written exactly at frontier positions and nowhere else.

Define the complete cleanup map

$$\mathcal{C}_2(u_i^{(2)}) = \left[\begin{array}{c} \widehat{X} \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ \hline (1 - \ell - \chi)e_{D_0} \\ 0 \\ 0 \end{array}\right]. \tag{55}$$

Hence all positional, record, frontier-memory, and temporary attention coordinates are erased after their information has been used. At positions that are neither frontier nor answer positions, the cleanup writes the nonzero inert role e_{D_0}; this guarantees that the subsequent normalization is well defined.

Every coordinate of $\mathcal{C}_2$ is affine or a linear combination of ReLUs of affine functions of the same residual input. In particular, both f and y^{ans} use only the reusable threshold ramps defined in Section C.2. Therefore $\mathcal{C}_2$ has the form of equation 13, and the generic residual-cleanup construction gives

$$u_i^{(2)} + \mathrm{FFN}_2(u_i^{(2)}) = \mathcal{C}_2(u_i^{(2)}) \tag{56}$$

exactly.

At a frontier position i_c, we have $\ell = 1$, $y^{\text{ans}} = 0$, and

$$f = \sum_{v \in F_{c+1}} e_v.$$

Hence the pre-normalization content is

$$\widehat{X} = e_\star + \sum_{v \in F_{c+1}} e_v.$$

After Euclidean normalization,

$$h^{(2)}_{i_c} = \frac{e_\star + \sum_{v \in F_{c+1}} e_v}{\sqrt{1 + |F_{c+1}|}} = \phi(F_{c+1}). \tag{57}$$

At the answer position, $\ell = 0$, $\chi = 1$, $f = 0$, and $y^{\text{ans}} = e_{c_\star}$. Since this vector already has unit norm,

$$h^{(2)}_{i_A} = e_{c_\star}. \tag{58}$$

Thus Block 2 first extracts expansion, history, and candidate information through attention, then converts these finite-softmax signals into exact indicators with a shared ReLU FFN, and finally uses residual cleanup and normalization to return either the canonical next-frontier state $\phi(F_{c+1})$ or the exact reachable candidate $e_{c_\star}$.

D Graph Experiments: Training and Baseline Performance

The experiments in Section 4 use GPT-2-style causal decoders with two or six Transformer layers, trained from scratch on simplified ProsQA (Hao et al., 2025). Nodes are represented by numerical identifiers. Each question specifies a directed graph, a root, and two candidate nodes, exactly one of which is reachable from the root. We follow the Coconut curriculum, progressively replacing explicit reasoning steps with up to four continuous thoughts. Training uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-2}, and a constant learning rate of 10^{-4}, without warmup or learning-rate decay.

Figure 8 reports native-task accuracy for the evaluated checkpoints. The original test questions have reachable targets at depths 3 or 4; this evaluation uses three latent positions for three-hop questions and four for four-hop questions. The two-layer and six-layer models achieve 98.0% and 61.1%, respectively. These depth-matched budgets differ from the fixed-budget evaluations below: the depth-targeted analysis in Section 4.1 uses four steps throughout, while the budget sweep evaluates each budget from zero to four on all original test questions.

Depth-targeted query construction. The original test questions query reachable targets at depths 3 or 4. To evaluate depth-selective readout at depths 1–4, we select the 264 test graphs that contain at least one reachable node at every depth in this range. For each graph, every node whose shortest-path distance from the root is $d \in \{1, 2, 3, 4\}$ is used as a reachable target. We pair it with the graph's original unreachable negative candidate and evaluate both candidate orders. Only the candidate pair is changed: the graph, root, edge order, prompt length, and negative candidate remain fixed. This procedure produces 1,474, 2,216, 1,916, and 960 queries at depths 1–4, respectively, for a total of 6,566 queries.

All queries are evaluated with four latent positions. Accuracy is determined by the model's argmax over its full vocabulary, rather than by a forced choice between the two candidates. For each depth, we first average over target nodes and candidate orders within each graph and then assign equal weight to the 264 graphs. The cosine-AUROC analysis uses unmodified original-question trajectories from the same graphs and follows the same graph-level averaging procedure.

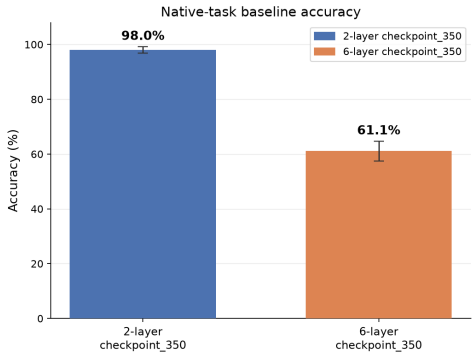

Figure 8: Native-task accuracy of the two-layer and six-layer checkpoints with a latent budget matched to the question's three- or four-hop depth. The original test set contains no one- or two-hop questions. Fixed-budget evaluations are reported separately below.

E Sensitivity to Model Depth and Latent Budget: Six-Layer Results

We additionally evaluated a six-layer checkpoint trained with the same curriculum. The depth-targeted results below use normal latent generation with a fixed budget of four latent tokens. On the original test set, the checkpoint achieved 61.6% full-vocabulary accuracy, compared with 73.9% for the two-layer checkpoint under the same fixed budget.

Target depth	All KV	P_1 only	P_2 only	P_3 only	P_4 only
1	15.1	23.3	15.6	15.2	15.0
2	21.7	28.3	21.6	21.3	21.6
3	30.9	37.4	30.4	31.0	30.6
4	58.2	51.9	57.9	59.3	59.3

Cosine AUROC of input vector z_t

Input vector	Depth 1	Depth 2	Depth 3	Depth 4
z_1	0.70	0.55	0.44	0.29
z_2	0.69	0.53	0.43	0.38
z_3	0.56	0.44	0.44	0.72
z_4	0.46	0.43	0.48	0.77

Table 9: Six-layer depth-targeted results using a fixed four-latent-token trajectory. Accuracy is computed over the full vocabulary. "All KV" denotes the normal answer readout, whereas P_i exposes only the indicated latent KV position. AUROC compares nodes at the specified BFS depth against nodes at other reachable depths.

The six-layer checkpoint did not reproduce the clear depth–position alignment observed in the two-layer model. Restricting answer readout to a single latent KV position produced only weak positional selectivity, while the input-vector AUROCs favor depth 1 for z_1, z_2 and depth 4 for z_3, z_4, rather than each input's corresponding depth.

Because this checkpoint also has substantially weaker held-out task performance, the comparison does not isolate the effect of model depth on frontier-like representations.

We also vary the latent budget on the original test questions.

The six-layer checkpoint achieves higher accuracy with zero latent steps (39.86% versus 6.92%) but gains less from increasing the budget to four (61.58% versus 73.87%). As in the zero-latent controls in Appendix B.5, dependence on latent recurrence varies with the model and evaluation setting. We conjecture that the deeper model implements BFS within its Transformer layers.

Latent tokens	2-layer (%)	6-layer (%)
0	6.92	39.86
1	39.74	51.67
2	44.99	60.86
3	72.67	61.22
4	73.87	61.58

Table 10: Accuracy under a fixed latent-token budget on the original ProsQA test questions.

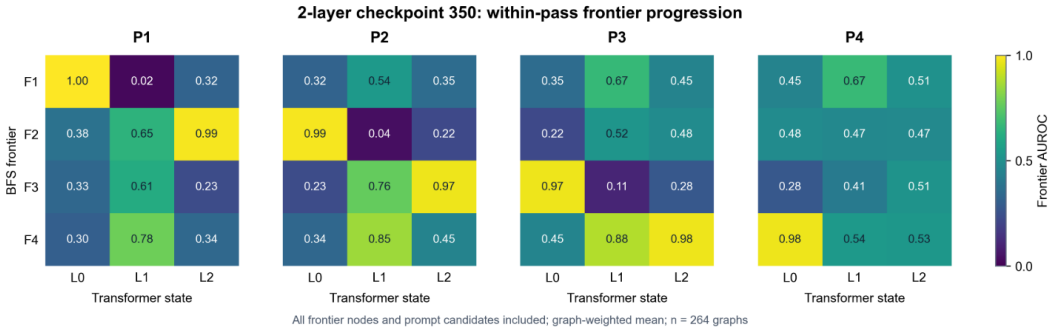

Figure 9: **Layer-wise evolution of frontier information.** Each panel corresponds to a latent position P_i, and each cell reports the frontier AUROC obtained from the hidden state at L0, L1, or L2. At P_1–P_3, the input state at L0 primarily represents the current frontier F_i, whereas the output of the second Transformer block at L2 primarily represents the next frontier F_{i+1}. Values are graph-weighted means over 264 graphs, with all frontier and prompt-candidate nodes included.

F Within-Pass Frontier Evolution and Answer Readout

To connect the depth alignment in Section 4.1 with the frontier update, we examine hidden states within each Transformer pass. At latent position P_i, we rank nodes by cosine similarity between the hidden state and their tied embeddings, and report AUROC for frontier F_j against all other reachable, non-root depths. L0 denotes the input to the first block; L1 and L2 denote the outputs of the first and second blocks.

At P_1–P_3, the input preferentially aligns with F_i, while the second-block output aligns with F_{i+1} (Figure 9). The L2 output is then fed back as the next position's L0 input. This pattern is consistent with advancing a frontier within one Transformer pass and carrying the result to the next latent position. It does not identify the exact operations performed by each block.

Access to P_4 at answer position	4-hop accuracy (%)
L1 and L2	**98.64**
L1 only	**96.50**
L2 only	46.80
Neither	48.40

Table 11: **Layer-specific readout of the P_4 KV cache at the answer position.** We selectively allow the answer token to attend to P_4 in the first Transformer block (L1), the second block (L2), both blocks, or neither block.

The timing of cache writes helps distinguish the feedback vector from its cached representation. Each block forms its K/V entries from that block's input. The final output at P_3 is therefore produced after the K/V entries at P_3 have been formed. The depth-four signal visible at P3-L2 is fed into P_4 and can then enter its first-block cache. This motivates testing which answer block uses access to P_4.

First-block access to P_4 alone preserves 96.50% four-hop accuracy, close to 98.64% with both blocks; second-block access alone gives 46.80% (Table 11). The learned reader thus relies mainly on first-block access, whereas the construction reads history in Block 2. The shared signature is frontier organization across feedback and history, not identical attention circuits.

G Paired Counterfactual History Intervention

Construction. We use the same 264 official test graphs containing at least four BFS frontiers and retain the 59 graphs that admit a one-edge-endpoint counterfactual. In each pair, an old node $v \in F_2$ is replaced by an originally unreachable sink u, while an unchanged edge from F_3 to v makes v enter only the counterfactual F_4. The edit preserves edge count and order, serialization length, root, query, candidates, answer, target depth, F_1, F_3, $\mathcal{N}(F_3)$, and frontier cardinality at the edited depth; both graphs remain acyclic. Hence the pair changes whether v has already been visited without changing the symbolic current frontier or its outgoing neighborhood.

We evaluate both candidate orders and both transplant directions, averaging candidate orders within each graph. The reported signed effect is positive when an intervention moves v in the direction predicted by the donor's history. During the main KV intervention, the recipient prompt, absolute positions, and clean z_3 are fixed, and only one latent cache position is replaced.

Clean-pair validation. Table 12 confirms both the intended manipulation and a departure from the construction's idealized residual cleanup. Removing v from F_2 lowers its z_2 score, and making it newly reachable at depth four raises its z_4 score. Although the paired graphs have identical symbolic F_3, their clean z_3 states already differ in the history-consistent direction. Thus the learned feedback is frontier-aligned but not a function of frontier membership alone.

State	Counterfactual − original	Correct direction
z_2	-0.19066	100.0% (negative)
z_3	$+0.02424$	89.8%
z_4	$+0.04417$	94.9%

Table 12: Paired change in the old node's cosine score along the clean trajectory.

Interventions and controls. Transplanting all-layer P_2 KV does not redirect the recipient z_4 according to donor history (Table 13). The effect is small and opposite to the predicted direction. K/V and layer separation locates a small history-consistent component in second-layer values, but it is dominated by the opposite first-layer value effect. In contrast, transplanting the paired clean z_3 strongly redirects z_4; adding donor P_2 KV barely changes that effect. A matched P_1 position control, zero and prompt-mean replacements, self-patching, and restoration rule out position-independent corruption and implementation error. Self-patch and restoration reproduce the clean output exactly (maximum difference 0).

Intervention	Signed cosine shift	Correct direction
P_2 KV, all layers	-0.001049	1.7%
P_2 K only, all layers	$+0.000040$	57.6%
P_2 V only, all layers	-0.001055	0.0%
P_2 KV, layer 1	-0.001142	0.0%
P_2 KV, layer 2	$+0.000093$	83.1%
P_2 V only, layer 1	-0.001123	0.0%
P_2 V only, layer 2	$+0.000068$	100.0%
P_1 KV position control	-0.000050	0.0%
Zero P_2 KV	-0.000516	3.4%
Prompt-mean P_2 KV	-0.000489	6.8%
Donor z_3 only	$+0.053862$	100.0%
Donor z_3 and P_2 KV	$+0.052814$	100.0%
Self-patch sham	0	—
Restore after transplant	0	—

Table 13: Bidirectionally pooled effects of the paired counterfactual interventions.

Interpretation. The trained checkpoint lies between two idealized representations. It differs from our construction because feedback retains update-relevant history beyond the current frontier, rather than being perfectly cleaned by the FFN. It also differs from the cumulative construction of Zhu et al. (2025): the depth-selective cosine results and attenuation of revisited nodes show that old reachable nodes are not preserved as an undifferentiated full visited set. The causal result therefore narrows the learned mechanism without negating the frontier signature or the theorem's existence claim.

H Checkpoint Provenance and Training Configuration

Checkpoints used in the feedback–history experiments. Table 14 records the exact analyzed weights and distinguishes checkpoints trained in this work from third-party releases. Our GPT-2 Coconut checkpoint was initialized from the `Onlydrinkwater/gsm8k-cot-gpt2` CoT-SFT release and then trained by us with the Coconut curriculum (Hao et al., 2025). Connor Dilgren and Sarah Wiegreffe released the Llama-Coconut, PrOntoQA, and ProsQA checkpoints (Dilgren & Wiegreffe, 2026). Xilin Wei et al. released the SIM-CoT checkpoint through the InternLM model repository (Wei et al., 2025), and Zhenyi Shen et al. released the CODI checkpoint through the `zen-E` account (Shen et al., 2025). We did not retrain these third-party checkpoints.

Model and task	Analyzed checkpoint	Provenance / backbone
GPT-2 Coconut, GSM8K	`checkpoint_18`	This work;
Llama-Coconut, GSM8K	`checkpoint_10`	Dilgren and Wiegreffe; Llama-3.2-1B-Instruct
SIM-CoT, GSM8K	`checkpoint_28`	Wei et al.; `internlm` release
CODI, GSM8K	`pytorch_model.bin`	Shen et al.; `zen-E` release
Explicit CoT, GSM8K	`gsm8k-cot-gpt2`	`Onlydrinkwater` Hugging Face release
Coconut, PrOntoQA	`checkpoint_36`	Dilgren and Wiegreffe
Coconut, ProsQA	`checkpoint_40`	Dilgren and Wiegreffe

Table 14: Provenance of the language and logical-reasoning checkpoints. Directory names are omitted; checkpoint identifiers match the supplementary-code layout.

Our GSM8K model uses the standard 12-layer GPT-2 architecture (hidden size 768, 12 attention heads, context length 1024, and dropout 0.1) and float32 training. Each curriculum stage replaces two explicit reasoning steps with continuous thoughts; three latent stages therefore give a six-position native budget. We use three epochs per stage, no cross-stage mixing, and reset the optimizer at stage transitions. AdamW uses learning rate 10^{-4}, weight decay 10^{-2}, batch size 32, and four-step gradient accumulation (effective batch 128). The run starts at curriculum epoch 3 because the initialization has already received CoT supervision, is configured through epoch 25, and the analyses use the epoch-18 checkpoint.

Symbolic graph checkpoints. We train the graph models from scratch on the simplified ProsQA data with the public Coconut graph-training pipeline (Zhu et al., 2026). Both checkpoints use a GPT-2-style decoder with a 40-token vocabulary, hidden size 768, eight attention heads, context length 1024, and dropout 0.1; they differ only in Transformer depth. Training uses one latent position per stage, a maximum of four latent stages, an initial 100-epoch prefix and 50 epochs per scheduled curriculum stage (so stage 0 spans epochs 1–150), 0.1 cross-stage sampling, and optimizer resets at stage transitions. AdamW uses learning rate 10^{-4}, weight decay 10^{-2}, per-device batch size 256 on two GPUs, and no gradient accumulation; arithmetic is float32. Both runs contain 350 epochs and use `checkpoint_350`. The paper's primary mechanistic analysis uses the two-layer checkpoint, and the depth-sensitivity appendix uses the six-layer checkpoint.

Transformer layers	2	6
Checkpoint	350	350
Training origin	this work	this work
Role	primary analysis	depth-sensitivity appendix

Table 15: Symbolic graph checkpoint inventory. Both models use the shared configuration above and differ only in Transformer depth.