UNROLLED FLOW MODELS FOR REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching enables language generation in few steps, but whether additional integration steps improve reasoning remains unclear. We prove that a flow parameterized by a two-layer Transformer can solve graph reachability, with the required number of integration steps increasing with the target's distance from the root. Yet, standard flow language models can fail to benefit from additional steps on reasoning tasks. We attribute this limitation to objectives that supervise each time point independently, without explicitly training successive steps to build on one another. To address this, we instead train through the model's own latent rollout over a randomly sampled subinterval of $[0, 1]$, decoding only at the endpoint. On ProsQA, this raises accuracy to 97% and enables performance to improve with additional integration steps. For the longer rollouts required by reasoning tasks such as Sudoku and Maze, retracting the latent state onto a sphere stabilizes the dynamics and yields substantial gains over baselines with more than three times as many parameters. Sampling multiple rollouts further improves performance when paired with a parameter-free selection score, although reliable selection remains challenging for longer answers. Together, these results establish a theoretical basis for reasoning with flows and show how rollout training, stable latent dynamics, and rollout selection help realize this capacity in practice.

1 INTRODUCTION

Autoregressive language models can spend more computation at inference by generating a longer chain of thought (Wei et al., 2022) or, more broadly, by scaling test-time compute (Snell et al., 2024). Flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo et al., 2025) offers a different way to produce discrete answers (Eijkelboom et al., 2024). Instead of predicting one token after another, flow models start from noise and repeatedly update a continuous state that is decoded into a token sequence. The number of updates can be chosen at inference time. This makes flow models an appealing candidate for adaptive test-time computation.

Additional integration steps do not inherently guarantee deeper reasoning. In generation tasks, a small number of steps is often enough, and recent work has pushed flow language models toward one- or few-step sampling by learning *flow maps* that make direct jumps along a trajectory (Lee et al., 2026; Roos et al., 2026; Davis et al., 2026; Yoo et al., 2026). For reasoning, additional steps are useful only if later updates can build on information produced by earlier ones.

Prior work in latent and recursive reasoning motivates this work. Coconut shows that autoregressive language models can update a continuous latent state instead of emitting every intermediate thought as text (Hao et al., 2026). On graph reachability, Zhu et al. (2025) show that repeated latent updates can maintain a superposition of reachable vertices and expand it one hop at a time. Recursive reasoners similarly solve structured tasks by applying a small, weight-tied network repeatedly to an internal state (Wang et al., 2025; Jolicoeur-Martineau, 2025). An Euler rollout of a flow has the same basic form: one network is applied again and again to a latent state. We investigate how these repeated applications cooperate.

To show what integration steps can compute, we construct explicit weights for a two-layer Transformer that defines a flow solving the graph reachability problem. Its Euler rollout reaches vertices one hop per step, so the number of steps needed to separate the candidates depends on the distance from the root to the answer. However, this existence result does not establish that training recovers the construction (Section 3.1).

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.

Standard training of flow models does not directly assess whether an update produces a state that later updates can leverage. To address this limitation, we propose *Unrolled Flow Model* (UFM), which trains through its own latent rollout using a single prediction loss at the final state. To reduce memory usage for long rollouts, we backpropagate this loss through only the last few steps (Section 4). We stabilize these rollouts using sphere retraction to control latent-state growth (Section 5). On ProsQA and Sudoku-Hard, UFM improves with additional steps, whereas evaluated flow baselines show little gain. Multiple rollouts further improve reliability, although selecting among them remains challenging for tasks with long answers, such as maze solving.

Our contributions are: (i) We give a constructive result showing how flow integration can implement graph search with the number of steps determined by the instance. (ii) We introduce training through a terminal rollout loss and show that it makes flow steps useful for reasoning in our experiments, while connecting the resulting dynamics to a superposition-based mechanism. (iii) We use sphere retraction to stabilize long rollouts and demonstrate stronger scaling with additional steps than substantially larger flow baselines on structured reasoning tasks.

2 Background

2.1 Conditional flow matching

Setup. Given a prompt c, the model generates an answer with L positions. A flow transforms noise at $t = 0$ into an answer representation at $t = 1$, using N Euler steps on a grid $0 = t_0 < \cdots < t_N = 1$.

Flow matching. Flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo et al., 2025) defines a path between noise $\boldsymbol{x}_0 \sim p_0 = \mathcal{N}(0, \boldsymbol{I})$ and data $\boldsymbol{x}_1 \sim p_1$. We use the linear interpolant

$$I_t := (1-t)\,\boldsymbol{x}_0 + t\,\boldsymbol{x}_1, \qquad \widehat{\boldsymbol{x}}_\theta(\boldsymbol{x}, t) \approx \mathbb{E}\big[\boldsymbol{x}_1 \mid I_t = \boldsymbol{x}\big]. \tag{1}$$

The denoiser $\widehat{\boldsymbol{x}}_\theta$ predicts the clean endpoint from a point on this path. Since $\boldsymbol{x}_1 - \boldsymbol{x}_0 = (\boldsymbol{x}_1 - I_t)/(1-t)$, its corresponding velocity field is

$$\boldsymbol{v}_\theta(\boldsymbol{x}, t) = \frac{\widehat{\boldsymbol{x}}_\theta(\boldsymbol{x}, t) - \boldsymbol{x}}{1-t}. \tag{2}$$

Integrating $\dot{\boldsymbol{z}}_t = \boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$ from $t = 0$ to $t = 1$ transports noise toward the data distribution. In practice, we approximate this integration with Euler steps:

$$\boldsymbol{z}_{t_{k+1}} = \boldsymbol{z}_{t_k} + (t_{k+1} - t_k)\,\boldsymbol{v}_\theta(\boldsymbol{z}_{t_k}, t_k). \tag{3}$$

Conditioning and training. The prompt tokens are concatenated with L answer slots and processed by a bidirectional Diffusion Transformer (DiT) (Vaswani et al., 2017; Peebles & Xie, 2023). The DiT receives the flow time t as a conditioning input, while the prompt remains clean throughout the rollout; only the answer slots evolve. The conditional denoiser is trained using:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{(c,\boldsymbol{x}_1),\,\boldsymbol{x}_0,\,t\sim\mathcal{U}[0,1]}\Big[\ell\big(\widehat{\boldsymbol{x}}_\theta(I_t, t, c),\ \boldsymbol{x}_1\big)\Big]. \tag{4}$$

Here, ℓ is the per-example loss. Thus, each training example supervises a denoising prediction at one sampled time; the model is never evaluated on a state produced by its own preceding update. We return to this distinction in Section 3.2.

2.2 Flow Language Models

Flow Language Models (FLMs) apply this formulation to discrete tokens. FLM (Lee et al., 2026) represents the clean answer as a stack of one-hot vectors, $\boldsymbol{x}_1 \in \{0,1\}^{L\times V}$. Its denoiser therefore predicts a distribution over the vocabulary at every answer position:

$$\widehat{\boldsymbol{x}}_\theta(\boldsymbol{x}, t, c) = \big(p^i(\cdot \mid I_t = \boldsymbol{x},\ c)\big)_{1\le i\le L}. \tag{5}$$

The model is trained with cross-entropy, and the evolving state $\boldsymbol{z}_t \in \mathbb{R}^{L\times V}$ is moved toward this predicted distribution at each Euler step. In our experiments, FLM refers to the multi-step flow model, rather than the distilled one-step flow-map variant introduced in the same work.

S-FLM. S-FLM (Deschenaux & Gulcehre, 2026) instead represents tokens with learned unit-norm embeddings on the hypersphere. It follows a geodesic path from noise toward these embeddings, rather than evolving a vocabulary-sized state as in FLM. Its denoiser still produces a distribution over the vocabulary; each sampling step follows a posterior-weighted average of geodesic directions toward the token embeddings. Both methods train a cross-entropy denoiser at one sampled point on their respective noise paths. S-FLM additionally uses a truncated, adaptive noise schedule. ELF (Hu et al., 2026) also defines the flow in a continuous token-embedding space.

FLM and S-FLM are our flow baselines. UFM also evolves a continuous latent state, but it predicts a latent target and applies the vocabulary decoder only once, at the endpoint. Its rollout is trained jointly with a terminal loss, as described in Section 4.

3 Integration Steps and Reasoning Depth

3.1 A Two-Layer Flow for Graph Reachability

We define the graph-reachability task and the flow construction used in the existence result. The theorem below shows that the construction solves the task; its Euler discretization will later show how many integration steps are needed. Notation is collected in Table 2.

Graph and prompt. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph with $n = |\mathcal{V}| \le n_{\max}$ vertices and $m = |\mathcal{E}|$ edges. Let $r \in \mathcal{V}$ be a designated starting vertex, called the root, and let c_1, c_2 be two candidate destinations. Exactly one candidate, denoted $c_{i^\star}$, is reachable from r; we write $c_{3-i^\star}$ for the other candidate.

We write $\mathrm{dist}(r, v)$ for the length of a shortest directed path from r to v, and $\mathcal{V}_k := \{v \in \mathcal{V} : \mathrm{dist}(r, v) \le k\}$ for the vertices reachable within k hops. The sequence $\mathcal{V}_0 \subseteq \mathcal{V}_1 \subseteq \cdots$ stops growing at the eccentricity of the root, $D := \max\{\mathrm{dist}(r, v) : v \text{ is reachable from } r\}$. Thus, $\mathrm{dist}(r, c_{i^\star}) \le D$.

We also write $\mathrm{W}_k(r, v)$ for the number of directed walks of length k from r to v.

Following Zhu et al. (2025), the prompt lists the directed edges and then asks which candidate is reachable (Figure 4 in the appendix):

$$c = (\langle \mathrm{s} \rangle, u_1, v_1, \langle \mathrm{e} \rangle, \ldots, u_m, v_m, \langle \mathrm{e} \rangle, \langle \mathrm{Q} \rangle, c_1, c_2, \langle \mathrm{R} \rangle, r, \langle \mathrm{A} \rangle).$$

The flow state $\boldsymbol{z}_t$ evolves at the position marked by $\langle \mathrm{A} \rangle$, where the final answer is decoded.

Representation. Let V be the vocabulary size and d_{te} the embedding dimension. Each token v has an embedding $\boldsymbol{\phi}_v \in \mathbb{R}^{d_{\mathrm{te}}}$. As in Zhu et al. (2025), we assume orthonormal token embeddings,

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}, \qquad \boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_V], \qquad d_{\mathrm{te}} \ge V.$$

This lets $\langle \boldsymbol{\phi}_v, \boldsymbol{z}_t \rangle$ recover the coefficient of vertex v in the state. This assumption applies solely to the construction; the trained models use standard embeddings.

The first layer stores each edge's source and target embeddings in separate buffers at its $\langle \mathrm{e} \rangle$ position, and the two candidate embeddings in buffers at $\langle \mathrm{A} \rangle$. It initializes the answer state as $\boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r$ (Lemmas 6 and 8).

Constructed velocity field. The second layer is a softmax-free linear-attention head (Definition 9). It reads the content slot of $\langle \mathrm{A} \rangle$ as its query and the cached source and target edge embeddings as its keys and values. Using these cached edge endpoints, it applies $\boldsymbol{A}_{\mathcal{G}} = \sum_{(u_j \to v_j) \in \mathcal{E}} \boldsymbol{\phi}_{v_j} \boldsymbol{\phi}_{u_j}^\top$, the adjacency operator of $\mathcal{G}$ in embedding coordinates. With a fixed gain $\lambda > 0$, the head induces

$$\dot{\boldsymbol{z}}_t = \lambda \boldsymbol{A}_{\mathcal{G}} \boldsymbol{z}_t, \qquad \boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r. \tag{6}$$

We use a linear head where Zhu et al. (2025) use softmax attention because it makes the velocity linear in the state, so the flow has a closed form. At time $t = 1$, the construction compares $\langle \boldsymbol{\phi}_{c_1}, \boldsymbol{z}_1 \rangle$ and $\langle \boldsymbol{\phi}_{c_2}, \boldsymbol{z}_1 \rangle$ and returns the candidate with the larger score. This is a candidate-restricted readout. The experiments use the stricter full-vocabulary decoder; we discuss that difference in Remark 11.

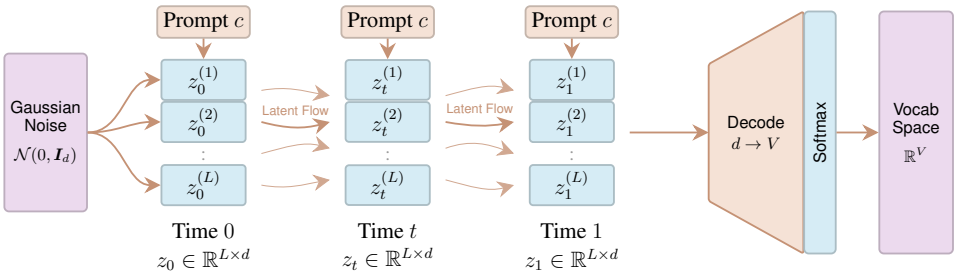


Figure 1: **UFM computes in latent space and decodes once.** At inference, a DiT repeatedly updates a noisy latent answer state from $t = 0$ to $t = 1$. A linear vocabulary decoder is applied only to the final state.

Exact flow The constructed velocity field has a closed-form solution. This gives the existence result: the flow separates the reachable candidate from the unreachable one on every instance.

Theorem 1 (Existence). *Fix a vocabulary of size V, $n_{\max}$, and $\lambda > 0$. There exists a two-layer Transformer, consisting of a prompt-caching layer followed by a linear-attention layer, whose weights are independent of the graph, root, and candidate pair. Its answer state satisfies Equation (6) for every directed graph with $|\mathcal{V}| \le n_{\max}$, every root r, and every candidate pair with exactly one reachable candidate $c_{i^\star}$. At time $t = 1$,*

$$\langle \boldsymbol{\phi}_v, \boldsymbol{z}_1 \rangle = \sum_{k \ge 0} \frac{\lambda^k}{k!} \mathrm{W}_k(r, v), \qquad v \in \mathcal{V}.$$

This coefficient is positive exactly when v is reachable from r. The candidate-restricted readout therefore returns $c_{i^\star}$, with the margin bound

$$\delta := \min_{0 \le k \le n_{\max}-1} \frac{\lambda^k}{k!} \le \langle \boldsymbol{\phi}_{c_{i^\star}}, \boldsymbol{z}_1 \rangle - \langle \boldsymbol{\phi}_{c_{3-i^\star}}, \boldsymbol{z}_1 \rangle.$$

Why the exact flow solves reachability. Since $\boldsymbol{A}_{\mathcal{G}} \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} = \boldsymbol{0}$, the solution is $\boldsymbol{z}_t = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + e^{\lambda t \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\phi}_r$. Orthonormality makes $\boldsymbol{A}_{\mathcal{G}}^k \boldsymbol{\phi}_r$ count length-k walks (Lemma 10), giving the theorem's nonnegative coefficient series. An unreachable vertex has no such walks; a reachable vertex has a shortest path of length $k^\star \le n_{\max} - 1$, contributing at least $\lambda^{k^\star}/k^\star! \ge \delta$ at $t = 1$. The network realization is established in Lemma 8 and Definition 9.

Unlike the Euler rollout considered next, the exact flow assigns positive mass to every reachable vertex at any $t > 0$. Reaching vertices one hop per step is therefore a property of the discretization, not of the flow itself.

Corollary 2 (Euler steps match graph distance). *For any Euler grid $0 = t_0 < \cdots < t_N = 1$ with positive step sizes,*

$$\boldsymbol{z}_{t_N} = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \prod_{i=0}^{N-1} \left(\boldsymbol{I} + \lambda \Delta t_i \boldsymbol{A}_{\mathcal{G}} \right) \boldsymbol{\phi}_r.$$

Hence, for every vertex v,

$$\langle \boldsymbol{\phi}_v, \boldsymbol{z}_{t_N} \rangle > 0 \quad \Longleftrightarrow \quad v \in \mathcal{V}_N.$$

Thus, the candidate-restricted readout is correct when $N \ge \mathrm{dist}(r, c_{i^\star})$. Taking $N \ge D = \mathrm{ecc}(r)$ suffices for every valid candidate pair with this root.

Expanding the product in Corollary 2 gives positive coefficients for $\boldsymbol{A}_{\mathcal{G}}^0, \ldots, \boldsymbol{A}_{\mathcal{G}}^N$. The state therefore contains contributions from walks of length at most N, and its vertex support is exactly $\mathcal{V}_N$. Step sizes change the weights on these walks and the number of steps determines how far information can propagate: a target farther from the root requires more Euler steps, using the same weights.

Remark 3 (Noisy initialization). *For $z_0 = \phi_{\langle A \rangle} + \phi_r + \sigma\xi$, with $\xi \sim \mathcal{N}(0, I)$, the exact time-1 flow returns the correct candidate with probability at least*

$$1 - \exp\left(-\frac{\delta^2}{4\sigma^2 e^{2\lambda\sqrt{m}}}\right).$$

The bound weakens with noise and graph size; see Section A.9.

The construction uses orthonormal embeddings, a softmax-free propagation head, a prompt cache, and a time-independent velocity field. It does not show that gradient descent recovers this solution. Trained UFM uses learned embeddings and reapplies its DiT at every step; Section 4.1 examines whether its dynamics show a related mechanism.

3.2 Integration Steps Alone Do Not Produce Reasoning

We test whether increasing the number of integration steps at inference improves reasoning accuracy in standard flow models.

Setup. As in Zhu et al. (2025), we use ProsQA (Hao et al., 2026), a directed graph-reachability task with two candidate answers. The models have two layers and approximately 15.7M parameters. The plotted UFM is trained with $N_{\text{train}} = 5$ steps and gradients through all five updates. UFM and S-FLM have approximately matched training budgets measured by the total number of model evaluations across examples; configurations are given in Section D.

Figure 2 shows test accuracy as the number of Euler steps N increases at inference. Both baseline curves are nearly flat: more integration steps add little accuracy. We revisit the same baselines on the harder Sudoku and Maze benchmarks in Section 5.3. On Sudoku-Hard, the 28.6M baselines change by only about one point from $N = 8$ to $N = 128$ (Table 6). Consequently, standard flow models require more than additional forward passes to improve reasoning.

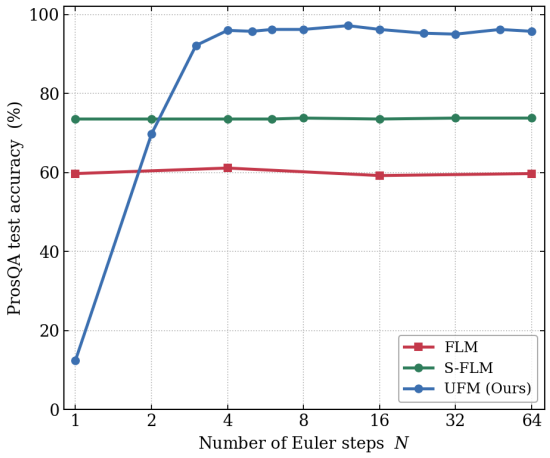


Figure 2: **Rollout training makes additional steps useful.** ProsQA test accuracy versus the number of Euler steps at inference. FLM and S-FLM remain nearly flat, while a single UFM trained with $N_{\text{train}} = 5$ improves from 12% at $N = 1$ to approximately 97% with longer rollouts. Decoding uses the full vocabulary.

This behavior is consistent with the standard flow objective in Equation (4). FLM and S-FLM train the denoiser to recover the answer from a ground-truth interpolant at a sampled time. The model is therefore not trained on the states created by its own preceding updates, and the objective does not assess whether one update leaves a useful state for the next. UFM instead trains a sequence of updates against one terminal answer loss.

S-FLM already evolves an embedding-space state; yet, its accuracy changes little with additional steps. Our construction motivates keeping the computation in latent space and decoding only at the endpoint; Section 4 trains these updates jointly through the rollout.

4 Unrolled Flow Model

We want early updates to produce states that later updates can use to reach the answer. We therefore train an *Unrolled Flow Model* (UFM) through its own rollout with a loss on the final answer. UFM keeps the answer state in a continuous latent space and decodes only once, at the endpoint.

Algorithm 1 Base UFM training on one batch. Task-specific variants are described in the text.

Require: Batch $(c, \boldsymbol{y})$, parameters θ, rollout length N_{train}, backpropagation length $1 \le N_{\text{back}} \le N_{\text{train}}$

Sample noise $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$ in the latent answer space
Sample a task-specific grid $0 \le t_0 < t_1 < \cdots < t_{N_{\text{train}}} = 1$
$\boldsymbol{z}_{t_0} \leftarrow (1 - t_0)\boldsymbol{\xi} + t_0 \,\text{Embed}(\boldsymbol{y})$
for $k = 0$ **to** $N_{\text{train}} - 1$ **do**
  $\Delta t_k \leftarrow t_{k+1} - t_k$
  $\widehat{\boldsymbol{x}}_\theta^{(k)} \leftarrow \widehat{\boldsymbol{x}}_\theta(\boldsymbol{z}_{t_k}, t_k, c)$
  $\boldsymbol{v}_\theta^{(k)} \leftarrow (\widehat{\boldsymbol{x}}_\theta^{(k)} - \boldsymbol{z}_{t_k})/(1 - t_k)$
  $\boldsymbol{z}_{t_{k+1}} \leftarrow \boldsymbol{z}_{t_k} + \Delta t_k \, \boldsymbol{v}_\theta^{(k)}$
  if $k < N_{\text{train}} - N_{\text{back}}$ **then**
    $\boldsymbol{z}_{t_{k+1}} \leftarrow \text{sg}(\boldsymbol{z}_{t_{k+1}})$ {stop gradients through early updates}
  end if
end for
$\hat{\boldsymbol{y}} \leftarrow \text{softmax}(\boldsymbol{z}_{t_{N_{\text{train}}}} \boldsymbol{W}_{\text{dec}}^\top)$ {decode once}
$\mathcal{L} \leftarrow \text{CE}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
return $\mathcal{L}, \nabla_\theta \mathcal{L}$

Latent rollout. $\boldsymbol{z}_t \in \mathbb{R}^{L \times d}$, where d is the hidden width. A DiT predicts a latent target, $\widehat{\boldsymbol{x}}_\theta(\boldsymbol{z}_t, t, c) \in \mathbb{R}^{L \times d}$, and defines the velocity as in Equation (2).

Unlike FLM and S-FLM, intermediate states are not decoded into distributions over the vocabulary. The vocabulary decoder is applied only to the final state,

$$\hat{\boldsymbol{y}} = \text{softmax}\big(\boldsymbol{z}_{t_N} \boldsymbol{W}_{\text{dec}}^\top\big), \qquad \boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{V \times d}.$$

We use softmax on ProsQA and replace it with StableMax (Prieto et al., 2025) on the longer Sudoku and Maze tasks. Details are in Section C.

Terminal rollout loss. We apply the loss only at the endpoint, so backpropagation trains successive updates through their combined effect on the final answer.

During training, the rollout starts from a partially noised answer at a sampled time t_0, $\boldsymbol{z}_{t_0} = (1 - t_0)\boldsymbol{\xi} + t_0 \,\text{Embed}(\boldsymbol{y})$ with $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$, and is integrated over a sampled subinterval ending at $t_N = 1$.

$$\mathcal{L}_{\text{UFM}}(\theta) = \mathbb{E}_{(c,\boldsymbol{y}),\, \boldsymbol{\xi},\, t_0} \left[\text{CE}\big(\text{softmax}(\boldsymbol{z}_{t_N} \boldsymbol{W}_{\text{dec}}^\top),\, \boldsymbol{y}\big) \right]. \tag{7}$$

No intermediate state receives a vocabulary loss. At inference, we set $t_0 = 0$, start from noise, and integrate the full interval $[0, 1]$. The complete long-rollout training and inference procedures, including sphere retraction and truncated backpropagation, are given in Section C.

4.1 What the Intermediate States Carry

Following Section 3.1, we inspect the logit $\langle \boldsymbol{z}_t, \boldsymbol{w}_v \rangle$ of each vertex v, where $\boldsymbol{w}_v$ is its output embedding. In Figure 3, non-reachable vertices remain near zero at every step, while several reachable vertices receive positive logits. At the first step, no single reachable vertex is clearly selected. The reachable group sits above the not-reachable one and drifts further right as t grows, and within it the optimal node separates the most.

While it does not establish that the trained model implements the construction, this qualitatively resembles Corollary 2, whose Euler states have support on all vertices reachable within k hops after k steps.

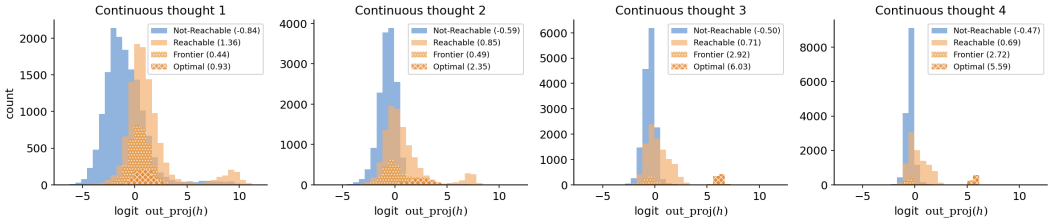


Figure 3: **Intermediate states separate reachable vertices.** We plot the logit $\langle \boldsymbol{z}_t, \boldsymbol{w}_v \rangle$ of vertex v, where $\boldsymbol{w}_v$ is row v of the final decoder $\boldsymbol{W}_{\text{dec}}$. Logits are centered separately for each problem by their mean over vertices. UFM trained with $N_{\text{train}} = 5$ is evaluated at $t = 0.2, 0.4, 0.6, 0.8$, on the 217 ProsQA test problems with $\text{dist}(r, c_{i^\star}) = 4$, averaged over four seeds. Vertices are grouped as *not reachable*, *reachable*, *frontier* ($\text{dist}(r, v) = k$), and *optimal* (the vertex at hop k on a shortest root-to-answer path).

5 Scaling to Long Rollouts

We evaluate UFM on Sudoku-Hard, Sudoku-Extreme, and Maze-Hard. These tasks require longer answer sequences and longer rollouts than ProsQA.

5.1 Benchmarks and evaluation protocol

Sudoku-Hard. We use the split of Deschenaux & Gulcehre (2026): 48k training puzzles and 2k reporting puzzles, each with 30 givens. UFM checkpoints are selected on an additional 2k validation split, disjoint from both sets. Our best FLM and S-FLM checkpoints are selected on the reporting set.

Sudoku-Extreme. Following Wang et al. (2025); Jolicoeur-Martineau (2025), training uses 1k base puzzles with 1000 augmentations each. We evaluate on the full test set of 422,786 puzzles. Checkpoint selection uses 2k additional puzzles from the original training pool, disjoint from the selected 1k base training puzzles and the test set. Each answer contains 81 positions.

Maze-Hard. We use the 30×30 benchmark of Wang et al. (2025); Jolicoeur-Martineau (2025), with 1000 training and 1000 test mazes whose shortest paths exceed 110 cells. The model predicts a 900-cell grid, of which roughly 110 cells lie on the solution path. UFM and both flow baselines select checkpoints on the test set; no separate validation split is used.

5.2 Training long rollouts

Sphere retraction. On long Sudoku rollouts, the latent norm can grow sharply (Section B). This changes the scale of the inputs to later updates and can destabilize the rollout. We control this growth on Sudoku and Maze with a position-wise sphere retraction,

$$\psi(\boldsymbol{z}) = \sqrt{d}\,\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}.$$

Let $\boldsymbol{z}_{t_k}$ denote the state read by the DiT and $y_{t_k} = \psi(\boldsymbol{z}_{t_k})$ its retracted version. The update is

$$\boldsymbol{z}_{t_{k+1}} = y_{t_k} + \frac{\Delta t_k}{1 - t_k}\big(\widehat{\boldsymbol{x}}_\theta(\boldsymbol{z}_{t_k}, t_k, c) - y_{t_k}\big), \qquad y_{t_{k+1}} = \psi(\boldsymbol{z}_{t_{k+1}}),\ \text{for } 1 \le k < N.$$

This is a projected Euler-type update: after the first step, its base is retracted, while the DiT continues to read the raw state. We use this retraction on Sudoku and Maze only. In a separate Sudoku ablation, training without it lowers accuracy; details are in Section B.

Truncated Backpropagation For the long tasks, we reduce activation memory by backpropagating only through the final $N_{\text{back}} = 6$ of $N_{\text{train}} = 24$ updates, as in Jolicoeur-Martineau (2025); Clark et al. (2024). The first 18 updates are computed without recording gradients and produce the state refined by the final six. We choose $N_{\text{back}} = 6$ to fit the available memory.

Table 1: Accuracy (%) on Sudoku-Hard. FLM and S-FLM are our runs of the authors' released code. Train is training-only wall-clock time on one NVIDIA A100. All methods use the same training data.

Method	Params	Train (h)	Sudoku-Hard
FLM (Lee et al., 2026)[a]	28.6M	1.6	51.9
S-FLM (Deschenaux & Gulcehre, 2026)[a]	28.6M	1.7	50.9[b]
UFM (ours)	8.4M	1.1	$\mathbf{86.9} \pm 0.9$

[a]Selected on the 2k reporting puzzles; UFM is selected on a separate 2k validation split. The UFM result is the mean and standard deviation over five independent noise seeds at $N_{\text{eval}} = 128$ on a uniform grid. [b]With ancestral noise removal at $N_{\text{eval}} = 128$; with the default greedy final step, S-FLM peaks at 50.2% at $N = 64$ (Table 6).

Each training example uses a newly sampled increasing time grid ending at $t = 1$, exposing the model to varying step sizes. The complete long-rollout procedures are given in Section C.

5.3 Comparison with Flow Baselines

We compare all methods at $N_{\text{eval}} = 128$ on Sudoku-Hard (Deschenaux & Gulcehre, 2026). FLM and S-FLM use the authors' released code and the best tested checkpoints and final-step settings at this budget (Section D.4).

On Sudoku-Hard, UFM reaches $86.9 \pm 0.9\%$, compared with 51.9% for FLM and 50.9% for S-FLM, using 8.4M rather than 28.6M parameters.

The gap widens on the harder tasks. On Sudoku-Extreme, UFM reaches $74.4 \pm 0.0\%$ with a uniform $N_{\text{eval}} = 128$ grid, averaged over five noise seeds. FLM and S-FLM reach 10.7% and 9.4%, respectively. On Maze-Hard, UFM reaches $89.3 \pm 0.5\%$, compared with 40.4% for FLM and 49.9% for S-FLM (Table 4). Published recursive-reasoner results and FRM are included for context in Tables 4 and 5.

5.4 Selecting Among Rollouts

Inference begins from Gaussian noise, so different rollouts can produce different answers. We therefore draw $K = 100$ rollouts and select one with a parameter-free score. *Margin* averages the gap between the two largest raw logits over output positions. It is an output-space analogue of δ in Theorem 1.

On mazes, scores average over 900 cells, although only roughly 110 describe the solution path. This makes selection harder, and selected accuracy remains below Pass@100. Detailed diagnostics are in Section D.1.

6 Related Work

Recursive and looped reasoning. Recursive reasoners repeatedly apply a small, weight-tied network to a latent state. HRM (Wang et al., 2025) uses slow and fast recurrent modules, while TRM (Jolicoeur-Martineau, 2025) uses a single recurrent block and an answer state. Subsequent work improves stability and scaling (Movahedi et al., 2026; Fein-Ashley & Rashidinejad, 2026; Gao et al., 2025). EqR and GRAM explore both depth, through more iterations, and breadth, through multiple trajectories (Huang et al., 2026; Baek et al., 2026). PTRM also studies sampling and selection (Sghaier et al., 2026).

Latent chain of thought. Coconut (Hao et al., 2026) feeds a language model's hidden state back as its next input embedding and introduces ProsQA. Zhu et al. (2025) show that a two-layer Transformer with continuous thoughts can solve graph reachability by maintaining a superposition of reachable vertices and expanding it over several steps. Their result provides the starting point for our construction in Section 3.1. UFM updates one latent answer state in place and uses answer-only

supervision throughout training. Coconut appends continuous thoughts and uses a curriculum that gradually replaces language reasoning steps.

Differentiating through generation. Backpropagating a terminal objective through a generative sampler is established in diffusion reward fine-tuning. DRaFT (Clark et al., 2024) backpropagates differentiable rewards through a diffusion sampler and introduces DRaFT-K, which truncates gradients to the final K sampling steps. UFM uses the same general optimization mechanism, but is trained from scratch with endpoint cross-entropy on reasoning tasks rather than fine-tuned against an external reward. Our N_{back} plays the role of a truncated sampler-backpropagation horizon.

Flow models for reasoning. Flow Reasoning Models (FRM) (Helbling et al., 2026) add recurrent self-conditioning to a flow model. Their Fixed-Point Forcing procedure uses a carry produced by the model's own rollout, while stopping gradients through that carry. They also report that increasing the number of sampling steps of a standard FLM does not improve Sudoku performance, consistent with our ProsQA observation in Section 3.2. FRM reports a peak accuracy of 99.5% on a fixed 1000-puzzle subset of the Sudoku-Extreme test set, included for context in Table 4. Its maze evaluation uses Maze-Unique rather than Maze-Hard.

Looped Flows (Suleymanzade et al., 2026), concurrent with this work, follows the TRM preprocessing and evaluation protocol on Sudoku-Extreme and Maze-Hard. It trains recurrent flow updates with local denoising losses and stops gradients between updates. UFM instead applies one terminal loss and backpropagates through the final N_{back} updates of the rollout.

7 Limitations

Our experiments use relatively small models, from 8M to 16M parameters, on synthetic structured reasoning tasks. Training through a rollout requires memory for the backpropagated suffix, which limited model size in this work. Extending UFM to larger models and less structured language-reasoning tasks will require more memory-efficient ways to train consecutive updates jointly.

The construction in Section 3.1 is an existence result. It relies on orthonormal embeddings and a linear propagation head, and does not show that gradient descent recovers this velocity field. The trained models share only a qualitative pattern with the construction, examined in Section 4.1.

Finally, a rollout begins from Gaussian noise, so a correct answer may appear among several trajectories without being selected by a single rollout. Our parameter-free selection rule recovers most of the Pass@100 gap on Sudoku-Extreme, but is less reliable for the much longer Maze-Hard outputs and could benefit from a learned correctness or halting head, as used in HRM and TRM (Wang et al., 2025; Jolicoeur-Martineau, 2025). Although UFM substantially improves over standard flow baselines, it remains behind the strongest reported recursive reasoners on Sudoku-Extreme.

8 Conclusion

We show how flow models can use additional integration steps to improve reasoning. On graph reachability, a two-layer construction propagates information along the graph, and its Euler discretization requires at least as many steps as the distance from the root to the answer. Standard flow language models do not use their steps in this way: their accuracy remains largely flat as the number of sampling steps increases. UFM changes this by training through the model's own latent rollout with a single loss at its endpoint. Accuracy then grows with the number of steps on ProsQA. For longer Sudoku and Maze rollouts, sphere retraction stabilizes the updates and allows an 8.4M-parameter model to outperform 28.6M standard-flow baselines. Multiple rollouts further improve reliability, although selecting among them is less effective for long answers.

Applying this approach to larger language models will require more memory-efficient rollout training and stronger selection mechanisms. Our results show that training through the rollout can make additional flow steps useful for reasoning.

AI USE STATEMENT

We used generative AI tools to improve writing, notation, and internal consistency; assist with intermediate mathematical calculations and proof steps; address gaps in the proofs and refine their presentation; and write and debug experimental code. We reviewed all AI-assisted work, tested the AI-assisted code, and verified every step of the proofs. We take full responsibility for the final content of this work, including all text, claims, proofs, results, code, and other artifacts produced with generative AI assistance.

ETHICS STATEMENT

This work studies reasoning with flow models on synthetic benchmarks: graph reachability (ProsQA), Sudoku, and maze solving. It involves no human subjects and no personal or sensitive data, and it releases no new datasets.

REPRODUCIBILITY STATEMENT

Section A states the assumptions of the construction and proves Theorem 1, Corollary 2, and Remark 3; Section B proves Proposition 12. Section C gives pseudocode for training and inference. Section D describes data splits, checkpoint selection, architecture, hyperparameters, and compute, including our baseline runs (Section D.4).

REFERENCES

Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. URL http://jmlr.org/papers/v26/23-1605.html.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning, 2026. URL https://arxiv.org/abs/2605.19376.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1vmSEVL19f.

Oscar Davis, Anastasiia Filippova, Pierre Ablin, Victor Turrisi, Amitis Shidani, Marco Cuturi, and Louis Béthune. Scaling categorical flow maps, 2026. URL https://arxiv.org/abs/2605.07820.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows, 2026. URL https://arxiv.org/abs/2605.11125.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 11735–11764. Curran Associates, Inc., 2024. doi: 10.52202/079017-0374. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/15b780350b302a1bf9a3bd273f5c15a4-Paper-Conference.pdf.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning, 2026. URL https://arxiv.org/abs/2605.12466.

Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model, 2025. URL https://arxiv.org/abs/2512.14693.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2026. URL https://arxiv.org/abs/2412.06769.

Alec Helbling, Andrey Bryutkin, Mauro Martino, Duen Horng Chau, Nima Dehmamy, and Hendrik Strobelt. Flow reasoning models: Turning flows into efficient recurrent reasoners, 2026. URL https://arxiv.org/abs/2606.29150.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows, 2026. URL https://arxiv.org/abs/2605.10938.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising, 2026. URL https://arxiv.org/abs/2602.16813.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026. URL https://arxiv.org/abs/2606.18206.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability, 2025. URL https://arxiv.org/abs/2501.04697.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model, 2026. URL https://arxiv.org/abs/2605.19943.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Ayhan Suleymanzade, Chanhyuk Lee, Floor Eijkelboom, Nicholas M. Boffi, İsmail İlkan Ceylan, and Jinwoo Kim. Thinking with looped flows, 2026. URL https://arxiv.org/abs/2609.11801.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

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 Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M. Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows, 2026. URL https://arxiv.org/abs/2607.00714.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought, 2025. URL https://arxiv.org/abs/2505.12514.

APPENDIX OVERVIEW

The appendix provides theoretical proofs, training and sampling algorithms, and additional experimental details and results.

- Appendix A gives the graph-reachability construction and proofs, including Euler discretization, noisy initialization, and full-vocabulary readout.
- Appendix B describes sphere retraction and reports the retraction ablation and norm diagnostic.
- Appendix C provides the training and inference algorithms for long rollouts.
- Appendix D gives the experimental settings, evaluation procedures, and additional results.

A GRAPH REACHABILITY: CONSTRUCTION AND PROOFS

We give the construction and proofs supporting Section 3.1. We describe the prompt representation and attention layers, then establish the exact-flow solution, its Euler discretization, and the noisy-initialization bound.

A.1 NOTATION AND ASSUMPTIONS

Notation is collected in Table 2. Fix a maximum graph size $n_{\max} \geq 2$ and a vocabulary containing the vertex labels and the special tokens. These special tokens are distinct from all vertex labels. Graphs are directed, may contain cycles, and have distinct edges, so $m \leq n_{\max}^2$.

Each token has an embedding $\boldsymbol{\phi}_v \in \mathbb{R}^{d_{\text{te}}}$. As in Zhu et al. (2025), the construction assumes $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}$, which requires $d_{\text{te}} \geq V$. These assumptions apply to the construction; the trained models use learned embeddings.

We set $\text{dist}(r, v) = \infty$ for unreachable vertices. The walk count includes zero-length walks: $\text{W}_0(u, v) = 1$ when $u = v$ and 0 otherwise. The evolving state $\boldsymbol{z}_t$ is the content slot of $\langle \text{A} \rangle$; the prompt information remains in the cache.

Table 2: Notation for Section 3.1 and this appendix. The last column gives the symbol of Zhu et al. (2025) where one exists; we rename their t_j and $\tilde{u}_v$ to free t for flow time and u, v for vertices.

Symbol	Meaning	Zhu et al.
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	directed graph, $n = \lvert\mathcal{V}\rvert \leq n_{\max}$, $m = \lvert\mathcal{E}\rvert \geq 0$ distinct edges	same
$(u_j \to v_j)$	source and target of edge j	(s_j, t_j)
r; c_1, c_2; $c_{i^\star}$, $c_{3-i^\star}$	root; candidates; reachable, unreachable candidate	same
$\text{dist}(\cdot, \cdot)$; $D = \text{ecc}(r)$	graph distance; eccentricity of the root	—
$\mathcal{V}_k$	vertices within k hops of r	$\mathcal{V}_c$
$\text{W}_k(u, v)$	number of directed walks of length k from u to v	—
$T = 3m + 7$; $T_{\max} = 3n_{\max}^2 + 7$	prompt length; its bound	—
$\langle \text{s} \rangle, \langle \text{e} \rangle, \langle \text{Q} \rangle, \langle \text{R} \rangle, \langle \text{A} \rangle$	special tokens	same
$\boldsymbol{\phi}_v \in \mathbb{R}^{d_{\text{te}}}$, $\boldsymbol{\Phi}$	token embedding, $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}$	$\tilde{u}_v$
$\text{ct}, \text{b}_1, \text{b}_2, \text{b}_3, \text{b}_4, \text{pe}$	residual-stream slots	content, buffers, pos
$\bar{\boldsymbol{p}}_a$; ε_{pe}	sinusoidal encoding of position a; separation gap	same; —
$\text{Pos}(\cdot)$	position in the prompt	same
$\text{Ch}(x, \ell)$; κ, β	attention chooser (token x, offset ℓ); its gains	—
ε_{th}	threshold	—
$\boldsymbol{z}_t \in \mathbb{R}^{d_{\text{te}}}$	content slot of $\langle \text{A} \rangle$ at flow time t	—
$\boldsymbol{A}_{\mathcal{G}}$; λ	propagation operator; value gain	—
$\Gamma_{u,v}$; δ	$\sum_k \lambda^k \text{W}_k(u, v)/k!$; readout margin	—
N	number of Euler steps	C (thoughts)
$\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$; σ	initialization noise; its scale	—
M_0	readout gain on the candidates	—

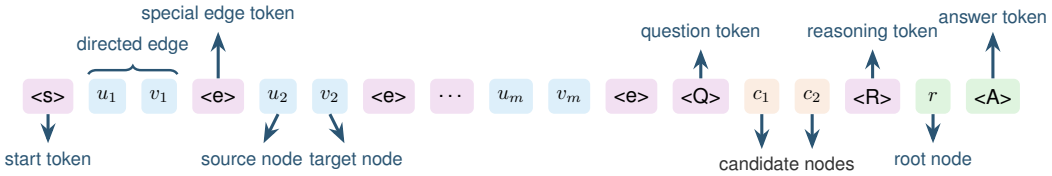


Figure 4: Visualization of the prompt token sequence.

A.2 Prompt format

A.3 Overview

Relation to Zhu et al. (2025). The first layer follows their construction: five attention choosers (Lemma 6) cache the information needed later, and a threshold MLP removes attention leakage. We adapt their construction as follows. (i) As in their paper, we state the construction with causal attention; Remark 7 shows that the same weights work with the bidirectional attention of our models. (ii) The last three heads read from $\langle \mathrm{A} \rangle$: two cache the candidates there, and the fifth writes the root r into the content slot of $\langle \mathrm{A} \rangle$, which sets the initial state of the flow. (iii) Their second layer, which expands the frontier by one hop per continuous thought with softmax attention and a threshold MLP, is replaced by a single linear attention head whose output defines the velocity field.

What is evaluated along the flow. Layer 1 is applied once to the prompt: it fills the buffers and sets the initial state $\boldsymbol{z}_0$ at $\langle \mathrm{A} \rangle$. At every time t, the velocity is the output of the propagation head with the current state $\boldsymbol{z}_t$ as query. Neither the chooser heads nor the threshold MLP act on $\boldsymbol{z}_t$, and the construction uses no normalization layers, so the velocity is linear in the state.

A.4 Transformer preliminaries

Prompt length. With $\langle \mathrm{A} \rangle$ appended, the prompt has length $T = 3m + 7$. Since the edges are distinct, $m \le n_{\max}^2$, so $T \le T_{\max} := 3n_{\max}^2 + 7$. All gains below are chosen from $T_{\max}$, so they do not depend on the graph.

Residual stream. We split the residual stream of width $d = 5d_{\mathrm{te}} + d_{\mathrm{pe}}$ into six slots: a content slot ct and four buffers $\mathrm{b}_1, \mathrm{b}_2, \mathrm{b}_3, \mathrm{b}_4$, all of width d_{te}, and a positional slot pe of width d_{pe}. Token embeddings live in the content slot and are orthonormal (Section A.1): $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_V]$ and $V = |\mathbb{V}|$.

In full-stream expressions, token embeddings are zero-padded outside the content slot. We keep positional encodings separate as $\boldsymbol{p}_i$; the displayed cached states omit this positional block.

Attention. One head of parameters $(\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O)$ computes the query $q_i = \boldsymbol{W}_Q(\boldsymbol{h}_i + \boldsymbol{p}_i)$, the keys $k_j = \boldsymbol{W}_K(\boldsymbol{h}_j + \boldsymbol{p}_j)$, and the attention scores

$$s_{i,j} = \frac{\exp\langle q_i, k_j \rangle}{\sum_{j'} \exp\langle q_i, k_{j'} \rangle},$$

where j' ranges over $j' \le i$ with causal masking and over all of $[T]$ in the bidirectional case (Remark 7).

Definition 4 (Sinusoidal Positional Encoding)**.** *Fix an even positional dimension $d_{\mathrm{pe}} \ge 2$, set $M = T_{\max} + 4$, and let $\omega = M^{-2/d_{\mathrm{pe}}}$. The positional encoding $\bar{\boldsymbol{p}}_i \in \mathbb{R}^{d_{\mathrm{pe}}}$ has entries:*

$$\bar{p}_{i,2k-1} = \cos(i\,\omega^k), \qquad \bar{p}_{i,2k} = \sin(i\,\omega^k), \qquad k \in [d_{\mathrm{pe}}/2],$$

and $\boldsymbol{p}_i$ is the vector living in the positional slot with $\mathrm{pe}(\boldsymbol{p}_i) = \bar{\boldsymbol{p}}_i$.

Lemma 5 (Separation, Zhu et al. 2025)**.** *With the choices in Definition 4, $\langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_i \rangle = d_{\mathrm{pe}}/2$ for all i, and there exists $\varepsilon_{\mathrm{pe}} > 0$, depending only on $T_{\max}$, such that $\langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_j \rangle \le d_{\mathrm{pe}}/2 - \varepsilon_{\mathrm{pe}}$ for all $i \ne j$ in*

$[T_{\max}+4]$. *Hence, for* $\ell \le 4$ *and* $\ell < i \le T$, $\langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_{j+\ell} \rangle$ *is maximized over* $j \in [T]$ *at the unique index* $j = i - \ell$, *with a gap of at least* $\varepsilon_{\mathrm{pe}}$.

Lemma 6 (Attention chooser, Zhu et al. 2025). *Consider a token* $x \in \mathbb{V}$, *an offset* $\ell \in \{1,2,3,4\}$ *and* $\varepsilon \in (0,1)$. *Under Definition 4, there exist* $\boldsymbol{W}_Q, \boldsymbol{W}_K$ *for a causal head, with gains depending only on* $T_{\max}$ *and* ε, *such that for any* $T \le T_{\max}$ *and any input sequence satisfying:*

(H1) $\boldsymbol{h}_i = \sum_{v \in \mathbb{V}} \alpha_{i,v}\, \boldsymbol{\phi}_v$ *with* $\alpha_{i,v} \ge 0$ *and* $\sum_v \alpha_{i,v}^2 = 1$, *for all* i;

(H2) $\langle \boldsymbol{\phi}_x, \boldsymbol{h}_i \rangle \in \{0,1\}$ *for all* i, *so each state is either equal to* $\boldsymbol{\phi}_x$ *or orthogonal to it;*

(H3) $\langle \boldsymbol{\phi}_x, \boldsymbol{h}_i \rangle = 0$ *for* $i \le \ell$, *so none of the first* ℓ *positions holds* x.

We have, for all $i \in [T]$,

$$s_{i,\,i-\ell} > 1 - \varepsilon \;\text{ if } \langle \boldsymbol{\phi}_x, \boldsymbol{h}_i \rangle = 1, \qquad s_{i,1} > 1 - \varepsilon \;\text{ otherwise.}$$

We write $\mathrm{Ch}(x, \ell)$ *for such a head.*

The idea is the following. Since the encoding is sinusoidal, $\bar{\boldsymbol{p}}_{j+\ell} = \boldsymbol{R}_\ell\, \bar{\boldsymbol{p}}_j$ for a fixed rotation $\boldsymbol{R}_\ell$, so a linear key can contain both $\bar{\boldsymbol{p}}_{j+\ell}$ and $\bar{\boldsymbol{p}}_j$. Taking $q_i \propto \big(\bar{\boldsymbol{p}}_i,\ \kappa \langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle\, \bar{\boldsymbol{p}}_1\big)$ and $k_j \propto \big(\boldsymbol{R}_\ell \bar{\boldsymbol{p}}_j,\ \bar{\boldsymbol{p}}_j\big)$ in two orthogonal subspaces, with $\boldsymbol{\phi}_{\bar{x}} = \sum_{v \ne x} \boldsymbol{\phi}_v$, gives

$$\langle q_i, k_j \rangle = \beta \Big(\langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_{j+\ell} \rangle + \kappa\, \langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle\, \langle \bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_j \rangle \Big).$$

If position i holds x, then $\langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle = 0$ and Lemma 5 puts the unique maximum at $j = i - \ell$ (and $i > \ell$ by (H3), so $i - \ell \ge 1$). Otherwise $\langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle \ge 1$ by (H1)–(H2), and for κ large the second term, uniquely maximized at $j = 1$, dominates. A large β turns both gaps into attention weights above $1 - \varepsilon$. We refer to Zhu et al. (2025) for the complete proof.

Our five heads use $x \in \{\langle \mathrm{e} \rangle, \langle \mathrm{A} \rangle\}$ and offsets $\ell \in \{1,2,3,4\}$. Each input position carries one token embedding, so (H1)–(H2) hold. Edge tokens occur at positions $3j + 1 \ge 4$, while the answer token occurs at position $3m + 7 \ge 7$. Since the edge heads use offsets at most two and the answer heads use offsets at most four, (H3) also holds.

Remark 7 (Bidirectional attention). *In a bidirectional head, the softmax runs over all* $j \in [T]$ *instead of* $j \le i$, *and the score is unchanged:*

$$\langle q_i, k_j \rangle = \beta \Big(\langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_{j+\ell} \rangle + \kappa\, \langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle\, \langle \bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_j \rangle \Big).$$

For a head that does not fire, $\langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle \ge 1$ *by (H1)–(H2), and by Lemma 5,* $\langle \bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_j \rangle$ *is uniquely maximized at* $j = 1$ *over all of* $[T]$, *not only over* $j \le i$; *for* κ *large, the head attends to* $\langle \mathrm{s} \rangle$. *For a head that fires,* $\langle \boldsymbol{\phi}_{\bar{x}}, \boldsymbol{h}_i \rangle = 0$ *and the score is* $\beta \langle \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_{j+\ell} \rangle$; *since* $j + \ell \le T + 4 \le T_{\max} + 4$ *for every* $j \in [T]$, *Lemma 5 gives the unique maximum at* $j = i - \ell$ *with the same gap* $\varepsilon_{\mathrm{pe}}$. *The softmax runs over at most* $T_{\max}$ *positions in both cases, so the gains* β, κ *of Lemma 6 are unchanged, and Lemma 8 holds. The propagation head is unaffected, since* $\langle \mathrm{A} \rangle$ *is the last position.*

A.5 Layer 1: caching

Lemma 8 (Static memory and answer initialization). *Append* $\langle \mathrm{A} \rangle$ *immediately after the root. One pass of the five chooser heads*

$$\mathrm{Ch}(\langle \mathrm{e} \rangle, 2), \quad \mathrm{Ch}(\langle \mathrm{e} \rangle, 1), \quad \mathrm{Ch}(\langle \mathrm{A} \rangle, 4), \quad \mathrm{Ch}(\langle \mathrm{A} \rangle, 3), \quad \mathrm{Ch}(\langle \mathrm{A} \rangle, 1),$$

followed by a threshold MLP, stores the source and target of each edge at its edge token, stores $\boldsymbol{\phi}_{c_1}$ *and* $\boldsymbol{\phi}_{c_2}$ *in separate buffers at* $\langle \mathrm{A} \rangle$, *and copies* $\boldsymbol{\phi}_r$ *into the content slot of* $\langle \mathrm{A} \rangle$. *The relevant states are*

$$\boldsymbol{h}_{\langle \mathrm{e} \rangle, j} = \begin{bmatrix} \boldsymbol{\phi}_{\langle \mathrm{e} \rangle} \\ \boldsymbol{\phi}_{u_j} \\ \boldsymbol{\phi}_{v_j} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{h}_{\langle \mathrm{A} \rangle} = \begin{bmatrix} \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{\phi}_{c_1} \\ \boldsymbol{\phi}_{c_2} \end{bmatrix}, \qquad \textit{so that} \qquad \boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r. \tag{8}$$

The key and value of the propagation head (Definition 9) read b_1 *and* b_2 *of* $\boldsymbol{h}_{\langle \mathrm{e}\rangle,j}$*, and* b_1 *is zero at every non-edge position.*

Non-firing heads assign more than $1-\varepsilon$ *attention to* $\langle \mathrm{s}\rangle$*, whose value is zero. We choose* $\varepsilon_{\mathrm{th}} = 1/(2\sqrt{n_{\max}})$ *and* $\varepsilon = \varepsilon_{\mathrm{th}}/2$*. The threshold and chooser gains can therefore be fixed using only* $n_{\max}$*.*

Proof. Each of the five heads is an attention chooser, so each one puts at least $1-\varepsilon$ of its attention on the position it is aimed at. We check that the offsets land correctly. With $\langle \mathrm{A}\rangle$ appended after the root, the positions are $\mathrm{Pos}(\langle \mathrm{s}\rangle) = 1$, $\mathrm{Pos}(u_j) = 3j-1$, $\mathrm{Pos}(v_j) = 3j$, $\mathrm{Pos}(\langle \mathrm{e}\rangle, j) = 3j+1$, $\mathrm{Pos}(\langle \mathrm{Q}\rangle) = 3m+2$, $\mathrm{Pos}(c_1) = 3m+3$, $\mathrm{Pos}(c_2) = 3m+4$, $\mathrm{Pos}(\langle \mathrm{R}\rangle) = 3m+5$, $\mathrm{Pos}(r) = 3m+6$ and $\mathrm{Pos}(\langle \mathrm{A}\rangle) = 3m+7$. So $\mathrm{Ch}(\langle \mathrm{e}\rangle, 2)$ and $\mathrm{Ch}(\langle \mathrm{e}\rangle, 1)$ read u_j and v_j; $\mathrm{Ch}(\langle \mathrm{A}\rangle, 4)$ and $\mathrm{Ch}(\langle \mathrm{A}\rangle, 3)$ read c_1 and c_2; and $\mathrm{Ch}(\langle \mathrm{A}\rangle, 1)$ reads r.

As in Zhu et al. (2025), all heads use the value matrix, $\boldsymbol{W}_V = [\, \boldsymbol{I}_{d_{\mathrm{te}}} - \boldsymbol{\phi}_{\langle \mathrm{s}\rangle}\boldsymbol{\phi}_{\langle \mathrm{s}\rangle}^{\top} \;\; \mathbf{0} \,]$, which will send $\boldsymbol{\phi}_{\langle \mathrm{s}\rangle}$ to zero.

Heads 1–4 write into buffers 1–4, respectively, and head 5 writes into the content slot. A non-firing head assigns more than $1-\varepsilon$ attention to $\langle \mathrm{s}\rangle$, whose value is zero. Each remaining output coefficient is therefore smaller than ε and is removed by the threshold MLP.

As in Zhu et al. (2025), a threshold MLP with threshold $\varepsilon_{\mathrm{th}}$, applied to each coefficient of the content slot and the buffers in the embedding basis, maps coefficients at most $\varepsilon_{\mathrm{th}}$ to 0 and coefficients at least $1-\varepsilon_{\mathrm{th}}$ to 1.

Each head puts attention above $1-\varepsilon$ on its target and leaks less than ε elsewhere, so with $\varepsilon < \varepsilon_{\mathrm{th}}$ every coefficient becomes exactly 0 or 1, and $\varepsilon_{\mathrm{th}}$ does not depend on the graph.

Only head 1 writes into buffer 1, and it fires only at edge tokens, so after thresholding buffer 1 is zero at every other position. Since buffer 1 is the key of the propagation head, only edge tokens contribute to its output.

Buffer 2 is written only by head 2, which fires at edge tokens, and buffers 3 and 4 only by heads 3 and 4, which fire at $\langle \mathrm{A}\rangle$. Hence at $\mathrm{Pos}(\langle \mathrm{A}\rangle)$ buffers 1 and 2 are empty, buffers 3 and 4 hold $\boldsymbol{\phi}_{c_1}$ and $\boldsymbol{\phi}_{c_2}$, and the content slot holds $\boldsymbol{\phi}_{\langle \mathrm{A}\rangle}$ together with the root written by head 5, so $\boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A}\rangle} + \boldsymbol{\phi}_r$, as claimed.

□

A.6 Layer 2: linear propagation

Definition 9 (Linear propagation head)**.** *The second layer is a single unnormalized linear-attention head (Katharopoulos et al., 2020), with*

$$\boldsymbol{W}_Q = [\, \boldsymbol{I}_{d_{\mathrm{te}}} \;\; \mathbf{0} \,], \quad \boldsymbol{W}_K = [\, \mathbf{0}_{d_{\mathrm{te}}} \;\; \boldsymbol{I}_{d_{\mathrm{te}}} \;\; \mathbf{0} \,], \quad \boldsymbol{W}_V = \lambda\, [\, \mathbf{0}_{d_{\mathrm{te}}} \;\; \mathbf{0}_{d_{\mathrm{te}}} \;\; \boldsymbol{I}_{d_{\mathrm{te}}} \;\; \mathbf{0} \,], \quad \boldsymbol{W}_O = [\, \boldsymbol{I}_{d_{\mathrm{te}}} \;\; \mathbf{0} \,]^{\top},$$

in a way that it reads the content slot as a query, the buffer 1 *as key and the buffer* 2 *as a value with a gain* λ*, and writes back on the content slot. There is no softmax and no normalizing denominator:*

$$o_i = \boldsymbol{W}_O \sum_{j \le i} \langle \boldsymbol{W}_Q \boldsymbol{h}_i,\, \boldsymbol{W}_K \boldsymbol{h}_j \rangle\, \boldsymbol{W}_V \boldsymbol{h}_j .$$

(the sum runs over all j *in the bidirectional case; at* $\langle \mathrm{A}\rangle$*, the last position, the two coincide)*

Define

$$\boldsymbol{A}_{\mathcal{G}} := \sum_{(u_j \to v_j) \in \mathcal{E}} \boldsymbol{\phi}_{v_j}\, \boldsymbol{\phi}_{u_j}^{\top}. \tag{9}$$

By Lemma 8, the buffer 1 is non-zero only at the edge tokens, where it contains $\boldsymbol{\phi}_{u_j}$ and the buffer 2 contains $\boldsymbol{\phi}_{v_j}$. Hence, with a query $\boldsymbol{z}$ at $\langle \mathrm{A}\rangle$, the head returns $\lambda \boldsymbol{A}_{\mathcal{G}}\, \boldsymbol{z}$. Moreover, the terms of the sum $\boldsymbol{\phi}_{v_j}\boldsymbol{\phi}_{u_j}^{\top}$ are orthonormal in the Frobenius sense for distinct edges, thus

$$\|\boldsymbol{A}_{\mathcal{G}}\|_2 \le \|\boldsymbol{A}_{\mathcal{G}}\|_F = \sqrt{m}.$$

A.7 Proof of Theorem 1

Because $\boldsymbol{\phi}_{\langle \mathrm{A} \rangle}$ is orthogonal to every vertex embedding,

$$\boldsymbol{A}_{\mathcal{G}}\, \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} = \mathbf{0}, \tag{10}$$

so under

$$\dot{\boldsymbol{z}}_t = \lambda \boldsymbol{A}_{\mathcal{G}} \boldsymbol{z}_t, \qquad \boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r, \qquad t \in [0, 1], \tag{11}$$

the answer-token identity remains fixed.

Writing

$$\boldsymbol{z}_t = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + a_t \tag{12}$$

is only a mathematical decomposition of the single residual state, not a separate architectural block. Its vertex component satisfies

$$\dot{a}_t = \lambda \boldsymbol{A}_{\mathcal{G}} a_t, \qquad a_0 = \boldsymbol{\phi}_r. \tag{13}$$

Lemma 10 (Reachability from the matrix exponential)**.** *Let $\mathrm{W}_k(r, v)$ be the number of directed walks of length k from r to v. Then $a_t = e^{\lambda t \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\phi}_r$ and, for every $v \in \mathcal{V}$,*

(i) $\langle \boldsymbol{\phi}_v, \boldsymbol{z}_t \rangle = \langle \boldsymbol{\phi}_v, a_t \rangle = \sum_{k \geq 0} \dfrac{(\lambda t)^k}{k!} \mathrm{W}_k(r, v) \geq 0$*;*

(ii) for $t > 0$, this coordinate is positive iff v is reachable from r, and is exactly 0 otherwise;

(iii) if v is reachable, then $\langle \boldsymbol{\phi}_v, \boldsymbol{z}_1 \rangle \geq \delta := \min\limits_{0 \leq k \leq n_{\max} - 1} \dfrac{\lambda^k}{k!} > 0$.

Proof. Orthonormality gives $\boldsymbol{A}_{\mathcal{G}}\, \boldsymbol{\phi}_x = \sum_{j : u_j = x} \boldsymbol{\phi}_{v_j}$, hence by induction $\boldsymbol{A}_{\mathcal{G}}^k \boldsymbol{\phi}_u = \sum_v \mathrm{W}_k(u, v) \boldsymbol{\phi}_v$ for every vertex u. Expanding the exponential proves (i). All terms are nonnegative, proving (ii). For reachable v, a shortest path has length $k^* = \mathrm{dist}(r, v) \leq n_{\max} - 1$ and $\mathrm{W}_{k^*}(r, v) \geq 1$, so its single series term proves (iii). The equality $\langle \boldsymbol{\phi}_v, \boldsymbol{z}_t \rangle = \langle \boldsymbol{\phi}_v, a_t \rangle$ follows from $\boldsymbol{\phi}_v \perp \boldsymbol{\phi}_{\langle \mathrm{A} \rangle}$. □

Proof of Theorem 1. By Lemma 8, layer 1 sets $\boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r$. By Definition 9, the velocity at $\langle \mathrm{A} \rangle$ is $\lambda \boldsymbol{A}_{\mathcal{G}} \boldsymbol{z}_t$, whose time-1 flow is $\boldsymbol{z}_1 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + e^{\lambda \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\phi}_r$ since $\boldsymbol{A}_{\mathcal{G}} \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} = \mathbf{0}$. Lemma 10(i)–(ii) give the coefficients and their signs; the reachable candidate has coefficient at least δ by (iii) and the unreachable one exactly 0 by (ii), which gives the $\arg\max$ and the margin. The weights depend only on $n_{\max}$, the fixed vocabulary, and λ, as required. □

A.8 Euler discretization

Proof of Corollary 2. Since $\boldsymbol{A}_{\mathcal{G}} \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} = \mathbf{0}$, Euler integration gives

$$\boldsymbol{z}_{t_N} = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \prod_{i=0}^{N-1} (\boldsymbol{I} + \lambda \Delta t_i \boldsymbol{A}_{\mathcal{G}}) \boldsymbol{\phi}_r, \qquad \Delta t_i = t_{i+1} - t_i > 0.$$

Expanding the product gives $\sum_{j=0}^{N} a_j \boldsymbol{A}_{\mathcal{G}}^j$, where $a_0 = 1$ and every $a_j > 0$: each coefficient is a sum of products of positive step sizes and powers of λ. By Lemma 10,

$$\langle \boldsymbol{\phi}_v, \boldsymbol{z}_{t_N} \rangle = \sum_{j=0}^{N} a_j \mathrm{W}_j(r, v).$$

This coefficient is positive exactly when a walk of length at most N connects r to v, equivalently when $\mathrm{dist}(r, v) \leq N$. The reachable candidate therefore has a strictly larger score once $N \geq \mathrm{dist}(r, c_{i^\star})$.

If $N < \mathrm{dist}(r, c_{i^\star})$, both candidate scores are zero, so the readout does not yet separate them.

□

A.9 NOISY INITIALIZATION

Proof of Remark 3. For the exact flow with $\boldsymbol{z}_0 = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + \boldsymbol{\phi}_r + \sigma\boldsymbol{\xi}$, where $\sigma > 0$ and $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$,

$$\boldsymbol{z}_t = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + e^{\lambda t \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\phi}_r + \sigma e^{\lambda t \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\xi}.$$

Let $g_v = \langle \boldsymbol{\phi}_v, \boldsymbol{\xi} \rangle$, which are i.i.d. $\mathcal{N}(0,1)$ by orthonormality, and $\Gamma_{v,w} = \sum_k \lambda^k \mathrm{W}_k(v,w)/k! \geq 0$. By Lemma 10, for every vertex w,

$$\langle \boldsymbol{\phi}_w, \boldsymbol{z}_1 \rangle = \Gamma_{r,w} + \sigma N_w, \qquad N_w = \sum_v g_v \Gamma_{v,w} \sim \mathcal{N}(0, \rho_w^2),$$

with $1 \leq \rho_w \leq e^{\lambda\sqrt{m}}$. Since every $\mathrm{W}_k(v,w) \geq 0$, the covariance of N_{c_1} and N_{c_2} is $\sum_v \Gamma_{v,c_1}\Gamma_{v,c_2} \geq 0$, so $\mathrm{Var}(N_{c_{3-i^\star}} - N_{c_{i^\star}}) \leq \rho_{c_1}^2 + \rho_{c_2}^2 \leq 2e^{2\lambda\sqrt{m}}$. The readout fails only if $\sigma(N_{c_{3-i^\star}} - N_{c_{i^\star}}) \geq \Gamma_{r,c_{i^\star}}$, and $\Gamma_{r,c_{i^\star}} \geq \delta$ by Lemma 10(iii). The tail bound $\Pr[Z \geq u] \leq e^{-u^2/2}$ for $Z \sim \mathcal{N}(0,1)$ and $u \geq 0$ gives

$$\Pr\left[\arg\max_{v \in \{c_1, c_2\}} \langle \boldsymbol{\phi}_v, \boldsymbol{z}_1 \rangle = c_{i^\star}\right] \geq 1 - \exp\left(-\frac{\delta^2}{4\sigma^2 e^{2\lambda\sqrt{m}}}\right).$$

The component of $\boldsymbol{\xi}$ orthogonal to the vertex span is frozen and invisible to the readout, and $\mathbb{E}[\boldsymbol{z}_t] = \boldsymbol{\phi}_{\langle \mathrm{A} \rangle} + e^{\lambda t \boldsymbol{A}_{\mathcal{G}}} \boldsymbol{\phi}_r$. □

A.10 READOUT OVER THE FULL VOCABULARY

Remark 11 (Full-vocabulary readout)**.** *Under the noiseless initialization of Theorem 1, a reachable vertex other than $c_{i^\star}$ can have a larger coefficient because the coefficients count walks. We use the candidate embeddings stored in buffers 3 and 4 to define*

$$S_v = \langle \boldsymbol{\phi}_v, \boldsymbol{z}_1 \rangle + M_0\big(\langle \boldsymbol{\phi}_v, \boldsymbol{\phi}_{c_1} \rangle + \langle \boldsymbol{\phi}_v, \boldsymbol{\phi}_{c_2} \rangle\big). \qquad v \in \mathbb{V}.$$

By orthonormality, the added term is M_0 on the two candidates and zero elsewhere. Every vertex coefficient is at most $e^{\lambda n_{\max}}$; the answer-token coefficient is one, and all other special-token coefficients are zero. Thus, choosing $M_0 = 2e^{\lambda n_{\max}}$ gives $S_{c_{i^\star}} \geq M_0 + \delta$, $S_{c_{3-i^\star}} = M_0$, and $S_v < M_0$ for every other token.

The full-vocabulary readout therefore returns $c_{i^\star}$ with margin at least δ. This is a fixed linear decoder on the content slot and candidate buffers, with M_0 depending only on $n_{\max}$ and λ. The velocity field is unchanged.

B SPHERE RETRACTION

B.1 LOCAL GEOMETRIC EFFECT

For a nonzero position vector, define $\psi(z) = Rz/\|z\|_2$, with $R = \sqrt{d}$. The following statement describes the local effect of this retraction.

Proposition 12 (Local effect of sphere retraction)**.** *Let $v(z,t)$ be continuously differentiable near the sphere $\|z\|_2 = R$ over a fixed interval $[0,T]$, with $T < 1$. For z on the sphere and sufficiently small Δt,*

$$\psi\big(z + \Delta t\, v(z,t)\big) = z + \Delta t\, P_{\hat{z}} v(z,t) + \mathcal{O}(\Delta t^2), \qquad P_{\hat{z}} = \boldsymbol{I} - \hat{z}\hat{z}^\top, \quad \hat{z} = z/R.$$

The associated velocity is tangent to the sphere.

Proof. For nonzero z,

$$D\psi(z) = \frac{R}{\|z\|_2}\left(\boldsymbol{I} - \frac{zz^\top}{\|z\|_2^2}\right).$$

Taylor expansion at a point with $\|z\|_2 = R$ gives the stated formula. Since $\langle z, P_{\hat{z}} v \rangle = 0$, the resulting continuous-time dynamics preserve the norm. □

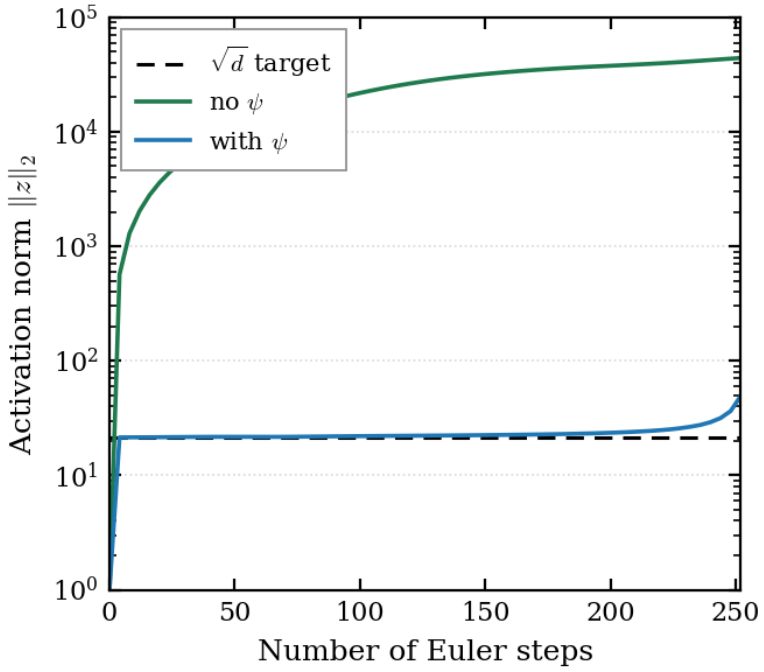


Figure 5: Raw latent norms in separately trained Sudoku-Extreme models with and without retraction. The dashed line marks $\sqrt{448}$, the norm of the retracted update base. The raw state itself need not remain on this sphere.

Relation to the implemented update. The algorithms in Section C evaluate the denoiser at the raw state and use its retracted version as the update base. If their difference is $\mathcal{O}(h)$, where h is the largest grid spacing, local Lipschitz continuity of the denoiser makes the prediction difference $\mathcal{O}(h)$. On an interval bounded away from $t = 1$, this changes an update by $\mathcal{O}(\Delta t_k h)$. The proposition therefore describes the leading-order interior dynamics under these conditions. It does not establish convergence through the endpoint $t = 1$.

B.2 Retraction ablation and norm diagnostic

We trained two Sudoku-Extreme models with and without ψ, using the same time-conditioned, constant-learning-rate recipe. At $N_{\text{eval}} = 64$, their best recorded test accuracies were 67% with retraction and 55% without it. Both models were trained separately. The ablation uses a different recipe from the cosine-schedule runs in the main comparison.

In the norm diagnostic (Figure 5), evaluated over a 252-step rollout, the model trained without retraction reaches a latent norm roughly $44{,}000$ times its initial scale. Retraction limits this growth, but does not hold the raw state at a fixed norm. The fixed radius applies to the retracted update base.

The construction in Section 3.1 also allows the latent norm to grow during propagation, while its final candidate decision is unchanged by positive rescaling.

C Long-Rollout Training and Inference

This appendix gives the full procedure used for the long Sudoku and Maze rollouts. Sphere retraction is applied independently to each answer position:

$$\psi(\boldsymbol{z}^{(i)}) = \sqrt{d}\,\frac{\boldsymbol{z}^{(i)}}{\|\boldsymbol{z}^{(i)}\|_2}, \qquad i = 1, \ldots, L.$$

Algorithm 2 Long-rollout UFM training on one batch with sphere retraction

Require: Batch $(c, \boldsymbol{y})$, parameters θ, rollout length N_{train}, backpropagation length $1 \le N_{\text{back}} \le N_{\text{train}}$, time-grid distribution q, noise scale σ, and retraction ψ

Sample noise $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$ in the latent answer space
Sample $0 \le t_0 < t_1 < \cdots < t_{N_{\text{train}}} = 1$ from q
$\boldsymbol{z}_{t_0} \leftarrow (1 - t_0)\sigma\boldsymbol{\xi} + t_0 \,\mathrm{Embed}(\boldsymbol{y})$
for $k = 0$ **to** $N_{\text{train}} - 1$ **do**
  Evaluate this update without gradient tracking when $k < N_{\text{train}} - N_{\text{back}}$
  $\Delta t_k \leftarrow t_{k+1} - t_k$
  $\widehat{\boldsymbol{x}}_\theta^{(k)} \leftarrow \widehat{\boldsymbol{x}}_\theta(\boldsymbol{z}_{t_k}, t_k, c)$
  if $t_{k+1} = 1$ **then**
    $\boldsymbol{z}_{t_{k+1}} \leftarrow \widehat{\boldsymbol{x}}_\theta^{(k)}$ {final endpoint prediction}
  else
    $y_{t_k} \leftarrow \boldsymbol{z}_{t_k}$
    if $k > 0$ **then**
      $y_{t_k} \leftarrow \psi(\boldsymbol{z}_{t_k})$
    end if
    $\boldsymbol{v}_\theta^{(k)} \leftarrow (\widehat{\boldsymbol{x}}_\theta^{(k)} - y_{t_k})/(1 - t_k)$
    $\boldsymbol{z}_{t_{k+1}} \leftarrow y_{t_k} + \Delta t_k \, \boldsymbol{v}_\theta^{(k)}$
  end if
  if $k < N_{\text{train}} - N_{\text{back}}$ **then**
    $\boldsymbol{z}_{t_{k+1}} \leftarrow \mathrm{sg}(\boldsymbol{z}_{t_{k+1}})$
  end if
end for
$\hat{\boldsymbol{y}} \leftarrow \mathrm{StableMax}\big(\mathrm{RMSNorm}(\boldsymbol{z}_{t_{N_{\text{train}}}})\boldsymbol{W}_{\text{dec}}^\top\big)$ {decode once}
$\mathcal{L} \leftarrow \mathrm{CE}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ {with the specified label smoothing}
return $\mathcal{L}, \nabla_\theta \mathcal{L}$

Algorithm 3 UFM inference with sphere retraction

Require: Prompt c, EMA parameters θ^{EMA}, evaluation grid $0 = t_0 < \cdots < t_{N_{\text{eval}}} = 1$, noise scale σ, and retraction ψ

Use θ^{EMA} for the denoiser and decoder
Sample $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I})$ in the latent answer space
$\boldsymbol{z}_{t_0} \leftarrow \sigma\boldsymbol{\xi}$
for $k = 0$ **to** $N_{\text{eval}} - 1$ **do**
  $\Delta t_k \leftarrow t_{k+1} - t_k$
  $\widehat{\boldsymbol{x}}_\theta^{(k)} \leftarrow \widehat{\boldsymbol{x}}_\theta(\boldsymbol{z}_{t_k}, t_k, c)$
  if $t_{k+1} = 1$ **then**
    $\boldsymbol{z}_{t_{k+1}} \leftarrow \widehat{\boldsymbol{x}}_\theta^{(k)}$ {final endpoint prediction}
  else
    $y_{t_k} \leftarrow \boldsymbol{z}_{t_k}$
    if $k > 0$ **then**
      $y_{t_k} \leftarrow \psi(\boldsymbol{z}_{t_k})$
    end if
    $\boldsymbol{v}_\theta^{(k)} \leftarrow (\widehat{\boldsymbol{x}}_\theta^{(k)} - y_{t_k})/(1 - t_k)$
    $\boldsymbol{z}_{t_{k+1}} \leftarrow y_{t_k} + \Delta t_k \, \boldsymbol{v}_\theta^{(k)}$
  end if
end for
$\hat{\boldsymbol{y}} \leftarrow \mathrm{StableMax}\big(\mathrm{RMSNorm}(\boldsymbol{z}_{t_{N_{\text{eval}}}})\boldsymbol{W}_{\text{dec}}^\top\big)$
return The highest-probability token at each position

The denoiser reads the raw state. After the first step, retraction is applied only to the update base. Noise scales and label smoothing are given in Table 7.

Table 3: UFM single-rollout accuracy with $N_{\text{eval}} = 128$ and a uniform time grid. Results are percentages, reported as mean $\pm$ standard deviation over five noise seeds.

Benchmark	Test examples	Pass@1 (%)
Sudoku-Extreme	422,786	74.4 ± 0.0
Sudoku-Hard	2,000	86.9 ± 0.9
Maze-Hard	1,000	89.3 ± 0.5

D Experimental Details

This appendix describes the data splits, training settings, and evaluation procedures used in our experiments.

Reporting convention. For the single-rollout results in Table 3, each noise seed gives one independently initialized rollout per test example, using a fixed checkpoint. We report the mean and standard deviation of exact-match accuracy over five noise seeds. Values are rounded to one decimal place, so a displayed standard deviation of 0.0 may be nonzero before rounding. The reported checkpoints have the DiT's time input disabled.

D.1 Selecting among rollouts

We evaluate $K = 100$ rollouts per problem on a 2000-puzzle subset of the Sudoku-Extreme test set and on all 1000 Maze-Hard test mazes. On the Sudoku subset, Pass@1 is 74.2%, Pass@100 is 98.65%, and margin-based selection reaches 98.6%. On Maze-Hard, margin-based selection reaches 92.0%, compared with 95.0% Pass@100.

We select among the K rollouts using the mean gap between the largest and second-largest raw logits,

$$\widehat{\delta} = \frac{1}{L} \sum_{i=1}^{L} \big(\ell_i^{(1)} - \ell_i^{(2)}\big).$$

Here, $\ell_i^{(1)}$ and $\ell_i^{(2)}$ are the two largest logits at position i, before probability normalization. This parameter-free score is an output-space analogue of the separation margin in Theorem 1.

Selection is less effective on Maze-Hard. A Sudoku answer has 81 positions, whereas a Maze answer contains 900 cells describing the maze layout and solution path. Averaging scores over cells shared by different candidate answers can dilute differences in their paths.

On a probe of 300 Maze boards with 50 rollouts each at N_{eval}=128, a subset of the boards in the evaluation above, 18 boards are recoverable, meaning that the first rollout is wrong but at least one of the 50 is correct; the remaining 282 boards are either already solved by the first rollout or solved by none of the 50. On these 18 boards, correct and incorrect rollouts differ in mean margin by 0.07, for mean margins around 200 in logit units, and the chooser picks a correct rollout on 9 of 18, against 38% for a uniformly random pick among the distinct answers.

These diagnostics suggest that the mean logit margin is less reliable for selecting long maze answers and a learned scorer, such as the latent reward model of GRAM (Baek et al., 2026), or the halting mechanism used in Jolicoeur-Martineau (2025); Wang et al. (2025) is a natural replacement.

D.2 ProsQA

FLM and UFM use a DiT backbone; S-FLM keeps the architecture of Deschenaux & Gulcehre (2026), which was designed for that method. All models have two layers and approximately 15.7M parameters and are trained on the same data. As in Zhu et al. (2025), we use the ProsQA questions whose answer is at most four hops from the root: 14,785 training, 257 validation, and 419 test questions, out of the 17,886 / 300 / 500 released. UFM is trained with $N_{\text{train}} = 5$ steps, so each example costs it 5 forward and backward passes. To approximately match the number of forward and backward passes, FLM and S-FLM see $5\times$ more training examples: 43,500 optimizer steps at batch

Table 4: Test accuracy (%) on Sudoku-Extreme and Maze-Hard with a single inference trajectory (Pass@1). Recursive-reasoner numbers are as reported in the respective papers; we did not rerun them. FLM and S-FLM are our runs of the authors' released code with the default greedy final step, at their best checkpoint and inference setting. UFM uses $N_{\text{eval}} = 128$ on a uniform grid; its results are means and standard deviations over five noise seeds, evaluated on 422,786 Sudoku-Extreme puzzles and 1000 Maze-Hard mazes. FRM's entry is its reported peak on a fixed 1000-puzzle Sudoku-Extreme test subset. Its maze evaluation uses Maze-Unique, so no Maze-Hard result is listed.

Model	Params	Sudoku-Extreme	Maze-Hard
Recursive reasoners			
Attractor Model (Fein-Ashley & Rashidinejad, 2026)	7M	54.3%	46.7%
HRM (Wang et al., 2025)	27M	55.0%	74.5%
URM (Gao et al., 2025)	14M	77.6%	–
TRM[a] (Jolicoeur-Martineau, 2025)	5M/7M	84.1%	85.3%
Attractor Model[b] (Fein-Ashley & Rashidinejad, 2026)	27M	91.4%	93.1%
EqR (D=64)[c] (Huang et al., 2026)	5M	93.0%	–
FPRM (Movahedi et al., 2026)	7M	94.2%	87.0%
Flow models			
FLM (Lee et al., 2026)	28.6M[†]	10.7%	40.4%
S-FLM (Deschenaux & Gulcehre, 2026)	28.6M[†]	9.4%	49.9%
UFM (ours)	8.4M	$74.4 \pm 0.0\%$	$89.3 \pm 0.5\%$
FRM (Helbling et al., 2026)	7M	99.5%[*]	–

[a] TRM-MLP (5M) on Sudoku, reproduced value from Huang et al. (2026); the original paper reports 87.4%. TRM-Att (7M) on Maze-Hard as reported. [b] Not reproduced by Movahedi et al. (2026), who obtain 71.4% on Sudoku. [c] D=64 outer iterations at test time, 4× the training budget; at D=16 the result is 86.4%. [†] At 8M, both baselines failed to train stably, reaching 1.5% on Sudoku-Extreme and then declining. [*] FRM reports this peak on a fixed 1000-puzzle *subset* of the Sudoku-Extreme test set, not the full 422,786-puzzle test set used for our Pass@1 evaluation.

Table 5: Test accuracy (%) on Sudoku-Extreme and Maze-Hard with multiple inference rollouts. *Rollouts* is the number of trajectories drawn at inference, followed by selection or voting. Recursive-reasoner numbers are as reported in the respective papers. UFM uses N_{eval}=128 Euler steps and K=100 rollouts with the chooser of Section 5.4. UFM is evaluated on a 2000-puzzle Sudoku-Extreme test subset and all 1000 Maze-Hard test mazes (Section D.1).

Model	Params	Rollouts	Sudoku-Extreme	Maze-Hard
Recursive reasoners				
GRAM[a] (Baek et al., 2026)	10M	20	97.0%	–
PTRM[b] (Sghaier et al., 2026)	5M/7M	100	98.75%	86.73%
EqR (D=64)[c,d] (Huang et al., 2026)	5M	128	99.8%	–
Flow models				
UFM (ours)	8.4M	100	98.6%	92.0%

[a] 20 parallel trajectories selected by a learned latent reward model. [b] Inference-only on a TRM checkpoint; best-Q@100 on Sudoku, majority vote on Maze; Pass@100 is 99.06% / 95.63%. [c] D=64 outer iterations at test time, 4× the training budget. [d] EqR reports 93.0% on Maze-Unique, a filtered variant, not Maze-Hard.

size 512 (≈22.3M examples), against 34,500 steps at batch size 128 for UFM (≈4.4M examples). UFM uses a smaller batch to fit the rollout in memory. Gradients flow through all five updates, but the gradient passed back through the latent state is scaled by 0.7 at each update; the forward value is unchanged.

D.3 Comparison with Recursive Reasoners

We also place UFM in context against published recursive reasoners. These rows are reported from the respective papers rather than reruns, so they provide a reference point rather than a controlled pairwise comparison.

D.4 Baseline replication on Sudoku-Hard

We use the best tested checkpoint of each baseline at 30k training steps, corresponding to 1.6 hours for FLM and 1.7 hours for S-FLM. For S-FLM, we used the truncated schedule in the authors' released code (the 43.2% setting in Table 1 of Deschenaux & Gulcehre (2026)), not the adaptive-schedule variant.

The released split has no validation set, so each baseline checkpoint is selected on the same 2k puzzles on which it is reported, which favors the baselines. The UFM checkpoint is instead selected on a separate 2k validation split (best exact match at N=64); this is the final checkpoint (19.7k steps, 1.1 h). The updated evaluation reaches $86.9 \pm 0.9\%$ at $N = 128$ over five noise seeds. Table 6 reports accuracy as a function of the number of sampling steps, with the default greedy final step. With ancestral noise removal at N=128, S-FLM reaches 50.9%, the number in Table 1; FLM has no such option. Training times exclude evaluation.

Table 6: Accuracy (%) on the hard Sudoku split of Deschenaux & Gulcehre (2026) as a function of the number of sampling steps N, with the default greedy final step and without ancestral noise removal. FLM and S-FLM are our runs of the authors' released code (28.6M parameters) with the cosine schedule of their code base, at their best checkpoint (30k steps), selected on these same puzzles, which favors the baselines. With ancestral noise removal at N=128, S-FLM reaches 50.9% (Table 1); with the released constant schedule it reaches 36.8%, and its authors report 43.2% with the truncated linear noise schedule and 45.0% with the additional adaptive schedule. UFM (8.4M parameters) is trained with 24-step rollouts and selected on a separate validation split. Training is training-only wall-clock time on one A100. The UFM row reports an earlier single-run step-count sweep. The updated five-noise-seed result at $N = 128$ is $86.9 \pm 0.9\%$ (Table 3). The second UFM row uses the same setup with a longer training budget of 22 hours. We report it separately from the main comparison in Table 1.

Model	Train (h)	N=8	16	32	64	128
FLM	1.6 h	51.2	51.8	51.8	51.6	51.9
S-FLM	1.7 h	48.9	49.5	50.0	50.2	50.1
UFM (ours)	1.1 h	2.8	65.1	80.2	84.4	86.1
UFM (ours)	22 h	64.1	97.0	99.4	99.8	99.8

D.5 Hyperparameters

UFM on Sudoku and Maze. All three benchmarks use a two-layer DiT with hidden width 448, eight attention heads, and 8.4M parameters. The backbone uses pre-norm RMSNorm, SwiGLU with expansion factor four, adaLN-Zero, two-dimensional rotary position embeddings, and 16 puzzle prefix tokens. The implementation supports sinusoidal time conditioning with frequency dimension 256. The reported checkpoints disable this input; flow time still enters through the update coefficients.

We use AdamW with $\beta = (0.9, 0.95)$, gradient clipping at 1.0, dropout 0.1, and an exponential moving average of the weights with decay 0.999. After warmup, the learning rate follows a cosine schedule from 10^{-4} to 10^{-5}. We retain weights and accumulation in FP32 and use BF16 matrix multiplications. Settings that differ across benchmarks are listed in Table 7.

Training grid and terminal loss. We train with $N_{\text{train}} = 24$ updates and backpropagate through the final $N_{\text{back}} = 6$; the preceding updates run without gradient tracking. For each example, we sample $t_0 \sim \mathcal{U}[0, t_{0,\max}]$ and initialize the state from the noise–answer interpolant. The upper bound $t_{0,\max}$ decreases from 0.6 to 0.2 over the first 30% of training. We sample and sort 23 further times in $(t_0, 1)$ independently for each example.

The final network call returns the predicted endpoint directly, equivalent to taking the last Euler update to $t = 1$. The decoder applies RMS normalization followed by an untied linear vocabulary projection. We compute one StableMax cross-entropy loss at this endpoint, with the label smoothing shown in Table 7.

Table 7: Benchmark-specific UFM training settings. The noise scale σ is the standard deviation of each latent noise coordinate. The Sudoku-Hard column describes the run used for the main comparison.

Setting	Sudoku-Extreme	Maze-Hard	Sudoku-Hard
Global batch size	768	128	256
Noise scale σ	$1/\sqrt{448}$	1	$1/\sqrt{448}$
Label smoothing	0.1	0	0.1
Weight decay	0.75	1.0	0.75
Warmup steps	2,000	2,000	500
Training budget (steps)	130k	125k	19.7k

D.6 Computing Resources

UFM training uses one NVIDIA A100 on Sudoku-Extreme and Sudoku-Hard, and two A100s on Maze-Hard. Training time excludes evaluation and is estimated from the median steady-state iteration rate in the logs.

On Sudoku-Hard, UFM takes 1.14 hours for 19.7k optimization steps. FLM and S-FLM take 1.07 and 1.12 hours, respectively, for 20k steps. Their selected 30k-step checkpoints require 1.61 and 1.68 hours of training. These estimates give the rounded training times reported in the main comparison.