DEPENDENCY-STRUCTURED GENERATION WITH MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive language models generate along a fixed left-to-right chain, whereas standard masked diffusion language models (MDLMs) train on randomly masked contexts, assigning equal training weight to all possible token reveal orders. Between these endpoints lies a structured regime: several tokens may be ready concurrently, while others should wait for specific evidence. We formalize this regime as dependency-structured generation, in which a model's reveal behavior is compactly described by a conceptual directed acyclic graph (DAG) over token positions, whose edges encode prerequisite relations. The framework recovers left-to-right autoregressive and standard (blockwise) MDLM objectives as boundary cases. The graph is a behavioral abstraction rather than a generated variable and is neither supplied nor constructed at inference. We instantiate this framework with MAPLE. During training, MAPLE uses a frozen teacher (e.g., a pretrained LLM) to extract a dependency graph for each target sequence; this graph induces a normalized training distribution over reveal orders. We derive an exact negative ELBO that jointly models the next reveal position and its token value. Under standard realizability, every global minimizer of the population negative-ELBO objective recovers the data sequence distribution. This optimality guarantee does not require the learned model to exactly reproduce the teacher's reveal behavior; the extracted dependencies instead serve as a finite-model inductive bias toward informative partial contexts. At inference, MAPLE retains the original parallel MDLM decoder unchanged. Continued pretraining of LLaDA 2 and DreamReasoner for mathematical reasoning and code generation improves task accuracy and substantially reduces teacher-dependency violations without sacrificing decoding parallelism.

1 INTRODUCTION

Token predictions often depend on specific evidence: a premise supports its conclusion, and a variable definition supports its later uses. Reveal order determines whether that evidence is visible. Autoregressive models fix a left-to-right order, whereas masked diffusion language models (MDLMs) train on partially masked sequences. Under the standard independent-masking objective with inverse-rate weighting, MDLM training is exactly an average over *uniformly* sampled token reveal orders (Sahoo et al., 2024; Hoogeboom et al., 2022; Ou et al., 2025; Nie et al., 2025). This flexibility covers arbitrary partial contexts, but gives the same weight to orders that expose useful evidence before a prediction and orders that do not. A finite model consequently spends capacity on poorly conditioned predictions. This leaves a natural middle ground between a single left-to-right chain and treating the entire reveal-order space symmetrically.

Identifying this middle ground is challenging. An L-token sequence admits $L!$ reveal orders; this combinatorial space has been highlighted as an optimization challenge for MDLMs (Ni et al., 2026). For sequences with prerequisite structure, however, plausible orders may occupy a much smaller, structured subset of these possibilities. Recent order-aware methods use planners, model confidence, teacher-estimated difficulty, or learned schedulers to implicitly steer training toward favorable regions of this space (Peng et al., 2026; Kim et al., 2026b; Chen et al., 2026; Hong et al., 2026b). Our approach instead makes the target subspace explicit: it uses a frozen teacher to compactly describe a plausible subset of reveal orders for each completed target sequence, then derives a principled training objective over that subspace.

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.

The compact description is a dependency graph associated with each completed sequence. Suppose a model can generate a sequence x through multiple reveal orders, and x contains a premise at position A and its conclusion at position B. If A is necessary evidence for B, a *well-structured* model should reveal A before B along every order it uses to generate x. The edge $A \to B$ summarizes this shared behavior. Collecting such relations yields a directed acyclic graph (DAG) $G(x)$. A reveal order is compatible with $G(x)$ if every prerequisite appears before its dependent; positions with no directed path between them may appear in either order. Thus one graph represents many reveal orders without enumerating them. A chain describes left-to-right revelation, an empty graph leaves the order unconstrained, and a blockwise DAG corresponds to blockwise generation (Arriola et al., 2025). We call this view *dependency-structured generation*. Here $G(x)$ is constructed after observing x only to describe reveal behavior; the generative model neither constructs nor consumes it at inference.

We turn this graph-based characterization into supervision with MAPLE, a **MA**sked **P**artial-order **L**anguage mod**E**l. For each completed training target, a frozen teacher estimates pairwise influence, from which we construct a sparse dependency graph offline. The teacher need not be a stronger sequence generator than the student because it annotates dependencies rather than supplying target token distributions. The graph induces a fixed, normalized training distribution over reveal orders. Treating the reveal order as latent, we derive an exact negative-ELBO objective for sequence likelihood that jointly trains the model to select the next masked position and predict its token. Training is tractable and has comparable cost to standard MDLM: a partially revealed state is constructed by iteratively sampling positions from the predefined graph-induced distribution, with no student or teacher forward passes, and the student is evaluated only once at the resulting state. Under model realizability, *every global minimizer of the population objective recovers the data sequence distribution*. As is common in latent-variable models, the teacher specifies the reveal-order space from which training states are drawn, while the optimal student model can adopt a different reveal distribution better suited to its own parameterization. At the objective level, the framework likewise recovers standard autoregressive, any-order MDLM, and block-autoregressive losses as special cases.

In practice, we derive variants of MAPLE without changing the standard MDLM architecture. A *hard* graph-induced policy strictly restricts training to teacher-compatible reveal orders, whereas a *soft* version broadens context coverage and allows a smooth fallback to the empty-graph case (i.e., the standard MDLM objective). Because standard MDLM decoding already uses token confidence to choose masked positions, MAPLE trains those same scores to prefer positions whose prerequisites are visible; no separate position-selection head is needed. The teacher and graph are used only during training, so inference retains the standard parallel MDLM decoder unchanged. Continued pretraining of LLaDA 2 (Bie et al., 2025) and DreamReasoner (Wu et al., 2026c) on mathematical reasoning and code improves task accuracy and reduces violations of teacher-estimated prerequisites.

2 A REVEAL-ORDER FORMULATION OF MDLMS

For the probabilistic formulation below, fix an observed prompt c and a supplied output length L; we suppress both from the notation. Let $x = (x_1, \ldots, x_L) \in \mathcal{V}^L$ be a completed output sequence, where each x_i is one token, $\mathcal{V}$ is the vocabulary, and $\mathcal{I} = \{1, \ldots, L\}$ is the position set. Let $O \subseteq \mathcal{I}$ denote the positions whose tokens have been revealed. The visible partial state $z(O, x)$ contains x_i at positions $i \in O$ and a mask elsewhere. For example, if $x = (A, B, C)$ and $O = \{1\}$, then $z(O, x) = (A, \texttt{[MASK]}, \texttt{[MASK]})$. Thus z determines both the revealed token values and the mask pattern. All model and data distributions below are implicitly conditional on (c, L).

2.1 PRELIMINARY: STANDARD MDLMS AS UNIFORM REVEAL-ORDER TRAINING

Let $\pi_{\theta,i}(\cdot \mid z)$ denote the MDLM's normalized distribution over vocabulary tokens at masked position i, conditioned on partial state z. Let $\mathfrak{S}_L$ be the set of all permutations of $\mathcal{I}$. For the standard MDLM objective, sample a reveal order $\gamma = (i_0, \ldots, i_{L-1}) \sim \mathrm{Unif}(\mathfrak{S}_L)$, so every permutation has probability $1/L!$, and define $O_k = \{i_0, \ldots, i_{k-1}\}$ and $z_k = z(O_k, x)$. The standard independently masked MDLM objective with inverse-mask-rate weighting (Sahoo et al., 2024; Nie et al., 2025)

admits the following exact any-order sequential representation (Ou et al., 2025, Theorem 2):

$$\mathcal{L}_{\mathrm{MDLM}}(x;\theta) := \mathbb{E}_{t\sim\mathrm{Unif}(0,1)}\mathbb{E}_{M\sim\mathrm{Bern}(t)^L}\left[\frac{1}{t}\sum_{i\in M} -\log \pi_{\theta,i}\big(x_i \mid z(\mathcal{I}\setminus M, x)\big)\right]$$
$$= \mathbb{E}_{\gamma\sim\mathrm{Unif}(\mathfrak{S}_L)}\sum_{k=0}^{L-1} -\log \pi_{\theta,i_k}(x_{i_k} \mid z_k) =: \mathcal{L}_{\mathrm{AO}}(x;\theta), \qquad \text{for every } \theta. \quad (1)$$

Here M is the masked-position set obtained by masking each position independently with probability t. Equation 1 uses the summed-token convention; division by L gives the usual mean-per-token normalization. We call $\mathcal{L}_{\mathrm{AO}}$ the *any-order* (AO) form because it averages equally over every reveal permutation (Hoogeboom et al., 2022). Appendix D gives the full derivation.

2.2 A NORMALIZED PARTIAL-STATE GENERATOR

The permutation-based form in Equation 1 has a one-step interpretation: at each partial state, choose one remaining masked position uniformly and predict its token. We generalize this transition by allowing state-dependent position probabilities. At a nonterminal state z, let $U(z)$ be its masked positions and let $\kappa_\theta(\cdot \mid z)$ be a probability distribution over $U(z)$. After selecting i, the model samples a token $y \in \mathcal{V}$ from $\pi_{\theta,i}(y \mid z)$. Their joint probability is

$$T_\theta(i, y \mid z) := \kappa_\theta(i \mid z)\, \pi_{\theta,i}(y \mid z). \quad (2)$$

Thus T_θ sums to one over all possible next position–token events. Sampling (i, y) and replacing the mask at i by y defines the next partial state. For example, from $z = (A, \texttt{[MASK]}, \texttt{[MASK]})$, the event $(i, y) = (3, C)$ has probability $\kappa_\theta(3 \mid z)\pi_{\theta,3}(C \mid z)$ and produces $(A, \texttt{[MASK]}, C)$. Repeating this transition defines a Markov process over partial states. The position selector may be fixed or learned; the standard any-order objective in Equation 1 uses uniform selection over $U(z)$.

Because T_θ is normalized, starting from the all-mask output state and applying the transition in Equation 2 L times defines a normalized joint distribution over a complete sequence x and reveal order $\gamma \in \mathfrak{S}_L$. Its joint probability and sequence marginal are

$$p_\theta(x, \gamma) := \prod_{k=0}^{L-1} T_\theta(i_k, x_{i_k} \mid z_k), \qquad p_\theta(x) := \sum_{\gamma'\in\mathfrak{S}_L} p_\theta(x, \gamma'). \quad (3)$$

Here $p_\theta(x, \gamma)$ is the joint model distribution, and $p_\theta(x)$ is its completed-sequence marginal. Section 3 next uses a teacher-derived dependency graph to define a training distribution over these reveal orders.

3 MAPLE: FRAMEWORK AND THEORY FOR GRAPH-INDUCED TRAINING

3.1 TEACHER-EXTRACTED DEPENDENCY GRAPH

For every completed training sequence x, a frozen teacher model extracts a sequence-specific dependency graph

$$G(x) = \big(\mathcal{I}, E(x)\big), \qquad E(x) \subseteq \mathcal{I}\times\mathcal{I}, \quad (4)$$

where $E(x)$ is its set of directed edges. An edge $j \to i$ declares position j a reveal prerequisite of position i. The extracted graph is held fixed during student training and must be a directed acyclic graph (DAG): a directed cycle would leave its tokens mutually blocked under the readiness rule below. Edges may point left-to-right ($j < i$) or right-to-left ($j > i$). Section 4.1 describes MAPLE's practical extractor. Below, we abbreviate the sequence-specific graph $G(x)$ as G.

Let $\mathrm{Anc}_G(i)$ contain every strict ancestor of position i, i.e., every position from which a directed path leads to i. At a revealed set O, define the graph-ready set

$$R_G(O) := \{i \in \mathcal{I}\setminus O : \mathrm{Anc}_G(i) \subseteq O\}. \quad (5)$$

Write $U(O) := \mathcal{I}\setminus O$ for the positions that remain masked. Thus $R_G(O) \subseteq U(O)$, and acyclicity guarantees that $R_G(O)$ is nonempty whenever $O \neq \mathcal{I}$.

Definition 1 (Graph-teacher variants)**.** *(i) Graph teacher.* For every completed sequence x and proper revealed set $O \subsetneq \mathcal{I}$, $h_G(\cdot \mid O, x)$ is any θ-independent probability distribution over $U(O)$.

(ii) Graph-hard teacher h_G^{hard}. It assigns zero probability to every non-ready position.

(iii) Uniform any-order teacher. $h_{\mathrm{AO}}(i \mid O) := \mathbf{1}\{i \in U(O)\}/|U(O)|$ is uniform over all remaining masked positions.

(iv) Uniform-ready teacher $\bar{h}_G$. It assigns equal probability to all ready positions:

$$\bar{h}_G(i \mid O, x) := \frac{\mathbf{1}\{i \in R_G(O)\}}{|R_G(O)|}. \tag{6a}$$

(v) Softened uniform-ready teacher $\tilde{h}_G^{\varepsilon}$. For $0 \le \varepsilon \le 1$, define

$$\tilde{h}_G^{\varepsilon}(i \mid O, x) := (1-\varepsilon)\bar{h}_G(i \mid O, x) + \varepsilon h_{\mathrm{AO}}(i \mid O), \tag{6b}$$

where $\tilde{h}_G^0 = \bar{h}_G$ and $\tilde{h}_G^1 = h_{\mathrm{AO}}$; for $\varepsilon > 0$, it has full support on $U(O)$.

Recall from Section 2 that $\gamma = (i_0, \ldots, i_{L-1})$ is a reveal order and $O_k = \{i_0, \ldots, i_{k-1}\}$ is its revealed set after k steps. Repeated sampling from h_G defines a normalized reveal-order distribution and a step-k distribution over revealed sets:

$$q_G(\gamma \mid x) := \prod_{k=0}^{L-1} h_G(i_k \mid O_k, x), \tag{7}$$

$$d_k^G(O \mid x) := \Pr_{\gamma \sim q_G(\cdot \mid x)}(O_k = O). \tag{8}$$

Different reveal orders γ can produce the same reveal set O after k steps, so $d_k^G(O \mid x)$ sums the probabilities of all such orders. A graph-hard teacher h_G^{hard} produces only reveal orders that respect every edge of G.

3.2 ELBO Objective and Efficient Training

The data contain only the completed sequence x; the reveal order γ is latent. For each γ, Equation 3 assigns joint probability $p_\theta(x, \gamma)$. We use $q_G(\cdot \mid x)$ from Equation 7 as a variational distribution over γ. Factoring the corresponding negative ELBO across successive position–token transitions gives the per-sequence training loss

$$\mathcal{L}_G(x;\theta) = \sum_{k=0}^{L-1} \mathop{\mathbb{E}}_{O_k \sim d_k^G(\cdot \mid x)} \left[\underbrace{\mathrm{KL}(h_G(\cdot \mid O_k, x) \,\|\, \kappa_\theta(\cdot \mid z_k))}_{\text{position selection}} + \underbrace{\mathop{\mathbb{E}}_{i \sim h_G(\cdot \mid O_k, x)} [-\log \pi_{\theta,i}(x_i \mid z_k)]}_{\text{token prediction}} \right]. \tag{9}$$

Proposition 1 (Negative-ELBO identity)**.** *For the reveal-order distribution q_G induced by a graph teacher h_G, if $p_\theta(x) > 0$, then*

$$\begin{aligned} \mathcal{L}_G(x;\theta) &= \mathbb{E}_{\gamma \sim q_G(\cdot \mid x)} [\log q_G(\gamma \mid x) - \log p_\theta(x, \gamma)] \\ &= -\log p_\theta(x) + \mathrm{KL}(q_G(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)) \ge -\log p_\theta(x), \end{aligned} \tag{10}$$

where $p_\theta(\gamma \mid x) := p_\theta(x, \gamma)/p_\theta(x)$ is the model's posterior over reveal orders.

The bound is tight exactly when $p_\theta(\cdot \mid x) = q_G(\cdot \mid x)$. Tightness is not required for sequence-marginal optimality (see Theorem 1). Appendix C gives the proof and covers zero-probability cases under the extended-real convention.

Efficient stochastic training. Exact evaluation of Equation 9 requires enumerating the revealed sets under d_k^G. We instead use a one-state Monte Carlo estimator $\widehat{\mathcal{L}}_G$ satisfying $\mathbb{E}[\widehat{\mathcal{L}}_G] = \mathcal{L}_G(x;\theta)$.

Draw $K \sim \mathrm{Unif}\{0, \dots, L-1\}$ and initialize $O = \varnothing$. Repeat K times: sample $i \sim h_G(\cdot \mid O, x)$ and update $O \leftarrow O \cup \{i\}$. With $z = z(O, x)$, one model forward pass computes

$$\widehat{\mathcal{L}}_G := |\mathcal{I}| \left[\mathrm{KL}(h_G(\cdot \mid O, x) \,\|\, \kappa_\theta(\cdot \mid z)) + \mathop{\mathbb{E}}_{i \sim h_G(\cdot \mid O, x)} [-\log \pi_{\theta,i}(x_i \mid z)] \right]. \tag{11}$$

After $K = k$ repetitions, the resulting set O is distributed as $d_k^G(\cdot \mid x)$, which proves the stated unbiasedness. For $\bar{h}_G$ and $\tilde{h}_G^\varepsilon$ (Definition 1), sampling O uses only the stored graph and random draws; neither the teacher nor the student is evaluated. The framework permits an independent position-selection head κ_θ; practical MAPLE instead sets κ_θ by normalizing a confidence function of π_θ (Section 4.2). The same model forward pass supplies the quantities needed for every remaining position, so both terms in Equation 11 are evaluated exactly, with no additional position sample or model forward.

3.3 Sequence-Marginal Optimality and Classical Endpoints

The data–teacher joint distribution ρ_G is sampled as follows. First draw $X \sim p_{\mathrm{data}}$ and retrieve its fixed graph annotation $G(X)$. Starting from $O_0 = \varnothing$, repeatedly sample $I_k \sim h_G(\cdot \mid O_k, X)$ and set $O_{k+1} = O_k \cup \{I_k\}$, yielding the teacher order $\gamma = (I_0, \dots, I_{L-1}) \sim q_G(\cdot \mid X)$. Thus $\rho_G(x, \gamma) := p_{\mathrm{data}}(x) q_G(\gamma \mid x)$, whose sequence marginal is p_{data}. At step k, the visible state is $Z_k = z(O_k, X)$ and the next position–token event is (I_k, X_{I_k}).

The model observes only Z_k, while q_G may depend on the complete X. We therefore marginalize over all completed sequences and teacher orders that reach the same visible state z. Because one position is revealed per step, the mask pattern in z uniquely determines k. For every z reached with positive probability under ρ_G, define the state-only target and apply $p(i, y \mid z) = p(i \mid z) p(y \mid i, z)$:

$$\begin{aligned} \overline{T}_G(i, y \mid z) &:= \rho_G(I_k = i, X_{I_k} = y \mid Z_k = z) \\ &= \underbrace{\rho_G(I_k = i \mid Z_k = z)}_{\kappa^{\mathrm{tar}}(i \mid z)} \underbrace{\rho_G(X_i = y \mid I_k = i, Z_k = z)}_{\pi_i^{\mathrm{tar}}(y \mid z)}. \end{aligned} \tag{12}$$

These are state-only targets obtained by the marginalization above; G is not an input to the student. Equation 2 has the same factorization. We call the model class *transition-realizable* if some θ_0 can achieve $\kappa_{\theta_0}(\cdot \mid z) = \kappa^{\mathrm{tar}}(\cdot \mid z)$ at every such state and $\pi_{\theta_0,i}(\cdot \mid z) = \pi_i^{\mathrm{tar}}(\cdot \mid z)$ whenever $\kappa^{\mathrm{tar}}(i \mid z) > 0$. Equivalently, $T_{\theta_0}(\cdot \mid z) = \overline{T}_G(\cdot \mid z)$ there. This is the standard population-level realizability idealization, made plausible by a sufficiently expressive Transformer with separate heads for κ and π. Let $\mathcal{J}_G(\theta) := \mathbb{E}_{X \sim p_{\mathrm{data}}}[\mathcal{L}_G(X; \theta)]$ and $\mathcal{J}_G^\star := \min_\vartheta \mathcal{J}_G(\vartheta)$.

Theorem 1 (Sequence-marginal optimality)**.** *For any fixed graph teacher h_G in Definition 1, if transition realizability holds, then, for every θ,*

$$\begin{aligned} &\mathrm{KL}(p_{\mathrm{data}}(x) \,\|\, p_\theta(x)) \le \mathcal{J}_G(\theta) - \mathcal{J}_G^\star, \\ &\theta^\star \in \arg\min_\vartheta \mathcal{J}_G(\vartheta) \Longrightarrow p_{\theta^\star}(x) = p_{\mathrm{data}}(x) \quad \textit{for all } x. \end{aligned} \tag{13}$$

Thus an excess population loss of δ bounds the sequence-modeling KL by δ, and every global minimizer recovers the data distribution without requiring $p_{\theta^\star}(\gamma \mid x) = q_G(\gamma \mid x)$. The full construction and proof appear in Appendix E.

Proposition 2 (Classical endpoint reductions)**.** *Let $G = G(c, L)$ be target-independent, so $\bar{h}_G(\cdot \mid O, x)$ is determined by the visible state z alone. Use the uniform-ready teacher $h_G = \bar{h}_G$ and fix the position selector, independently of the token predictor, as $\kappa_\theta(\cdot \mid z(O, x)) = \bar{h}_G(\cdot \mid O, x)$. Then Equation 9 reduces to: (i) the left-to-right autoregressive negative log-likelihood for a chain; (ii) the standard any-order MDLM objective (Hoogeboom et al., 2022; Sahoo et al., 2024) for the empty graph; and (iii) the standard blockwise MDLM objective (Arriola et al., 2025) for a block-chain DAG, with no within-block edges and all-to-all edges between successive blocks.*

Appendix D gives the full reductions and architectural qualifications.

4 MAPLE: Practical Adaptations

Practical MAPLE makes two choices beyond Section 3: how to extract $G(x)$, and how to transfer its position preferences into the confidence-derived position distribution κ_θ of a standard MDLM.

4.1 Predictive Dependencies and Offline Extraction

We call an edge $j \to i$ *predictive* when making x_j visible to $\pi_{\theta,i}$ provides useful evidence for predicting x_i. A graph made of predictive edges can ease model optimization by focusing training on informative partial contexts; noisy or uninformative edges weaken this benefit. MAPLE constructs $G(x)$ offline using a pretrained autoregressive teacher. For each target x_i, we score candidate tokens x_j with $j < i$ in two ways: *(i) attention-based*, which aggregates the teacher's attention from the prediction query for x_i to x_j; and *(ii) counterfactual-based*, which measures how much the teacher's log-probability of x_i decreases when x_j is ablated from its context. We sparsify these scores to form $G(x)$, which is preprocessed before training. Appendix H provides the extraction details.

4.2 Adapting MAPLE to Confidence-Based Decoding

We target MAPLE at MDLMs that use confidence-based decoding. Rather than add a separate position-selection head, we compute κ_θ by applying a fixed confidence function to the token predictions π_θ. This preserves exactly the architecture and decoding algorithm of MDLMs such as LLaDA and Dream (Nie et al., 2025; Ye et al., 2025). For $\varepsilon < 1$, using $\tilde{h}^{\varepsilon}_G$ as the target of the position-selection KL in Equation 9 would discourage high confidence on non-ready positions, but also push ready positions toward equal confidence, potentially hurting decoding quality. We therefore modify this KL target to preserve the model's relative confidence among ready positions.

Let $\bar{\kappa}_\theta := \mathrm{sg}[\kappa_\theta]$, where sg denotes stop-gradient, and assume it has full support on $U(O)$. Define

$$w_{\varepsilon,G}(i \mid O) := \varepsilon + (1-\varepsilon)\mathbf{1}\{i \in R_G(O)\}, \qquad i \in U(O),$$
$$h^{\mathrm{proj},\varepsilon}_{\theta,G}(i \mid O, z) := \frac{w_{\varepsilon,G}(i \mid O)\bar{\kappa}_\theta(i \mid z)}{\sum_{j \in U(O)} w_{\varepsilon,G}(j \mid O)\bar{\kappa}_\theta(j \mid z)}, \qquad 0 \le \varepsilon \le 1. \tag{14}$$

At $\varepsilon = 0$, the target assigns zero probability to non-ready positions and renormalizes the stop-gradient confidence scores over $R_G(O)$. Its KL to κ_θ equals $-\log \sum_{i \in R_G(O)} \kappa_\theta(i \mid z)$. Minimizing this KL increases the total probability assigned to $R_G(O)$ while preserving the relative confidence among ready positions. At $\varepsilon = 1$, the target equals $\bar{\kappa}_\theta$, so $\mathrm{KL}(h^{\mathrm{proj},1}_{\theta,G} \| \kappa_\theta) \equiv 0$ with zero gradient.

Let $d^{\varepsilon,G}_k$ denote the step-k revealed-set distribution induced by $\tilde{h}^{\varepsilon}_G$, as in Equations 7–8. Using $\tilde{h}^{\varepsilon}_G$ for revealed-set sampling and the token-loss expectation, and $h^{\mathrm{proj},\varepsilon}_{\theta,G}$ for the position-selection KL, the practical loss is

$$\mathcal{L}^{\mathrm{proj}}_{\varepsilon,G}(x;\theta) := \sum_{k=0}^{L-1} \mathop{\mathbb{E}}_{O_k \sim d^{\varepsilon,G}_k(\cdot|x)} \left[\mathrm{KL}\Big(h^{\mathrm{proj},\varepsilon}_{\theta,G} \| \kappa_\theta\Big) + \mathop{\mathbb{E}}_{i \sim \tilde{h}^{\varepsilon}_G} [-\log \pi_{\theta,i}(x_i \mid z_k)] \right]. \tag{15}$$

Equation 15 is not an exact ELBO because its position-selection KL and token expectation use different targets. It nevertheless has the exact fallback below and performs well empirically (Section 6). Training follows the same one-state procedure as Section 3.2, with the substitutions shown above.

Proposition 3 (Exact confidence-coupled fallback)**.** *Let κ_θ be a differentiable, strictly positive normalization of the same token-confidence scores used for decoding. If G is empty for any ε, or if $\varepsilon = 1$ for any G, then, for every θ, $\tilde{h}^{\varepsilon}_G = h_{\mathrm{AO}}$ and $h^{\mathrm{proj},\varepsilon}_{\theta,G} = \mathrm{sg}[\kappa_\theta]$. Consequently, $\mathcal{L}^{\mathrm{proj}}_{\varepsilon,G}(x;\theta) = \mathcal{L}_{\mathrm{AO}}(x;\theta)$ and their training gradients agree under stop-gradient. With the same π_θ and confidence-based commit rule, inference uses exactly the standard parallel MDLM procedure.*

Appendix G proves the projection and fallback claims.

5 Related Work

Learning and planning reveal order. Token order materially affects masked-diffusion training and decoding (Kim et al., 2025a; Garg et al., 2025). The OeMDM framework and its LoMDM instantiation (Hong et al., 2026b) provide a scheduler-based view that includes MDLM, autoregressive, and blockwise generation, while jointly learning full-sequence and partial-state schedulers. Fox et al. (2026) variationally learn an order and its inference-time selector; Adaptive Order Policies (Mohamud et al., 2026) and UPO (Hong et al., 2026a) train lightweight policies from correctness

or reward; Where-to-Unmask (Asano et al., 2026) learns an oracle-guided ranking planner; and Path Planning (Peng et al., 2025) performs inference-time planning with optional remasking. These approaches retain a learned or explicit selection mechanism during generation. MAPLE instead uses an offline prerequisite graph to supervise the existing MDLM and discards the graph at deployment.

Order-aware training. Methods without an additional inference module include PAPL (Peng et al., 2026), PUMA (Kim et al., 2026b), and dParallel (Chen et al., 2025b), which align training with confidence-based trajectories; Efficient-DLM (Fu et al., 2026) and AGDO (Deng et al., 2026), which use position- or attention-derived schedules; and TRIMS (Chen et al., 2026), which transfers an autoregressive teacher's scalar token difficulty. Their principal practical order signals are per-position scores. MAPLE instead supervises pairwise, sequence-specific prerequisites: readiness depends on which prerequisite tokens are visible, while unrelated positions remain unordered. Its contribution is this relational training signal together with graph-free confidence decoding, rather than order awareness or endpoint unification alone.

Dependencies in diffusion decoding. DAWN (Luo et al., 2026) and DAPD (Kim et al., 2026a) derive attention-based dependencies online, while DEMASK (Ringel et al., 2026) learns a pairwise predictor; all three use the resulting guidance during inference. Inference-only decoders UNCODE (Huang et al., 2026), KLASS (Kim et al., 2025b), and CCD (Chen et al., 2025a) instead modify reveal priorities without retraining; we evaluate them as decoding baselines. DA-DLM (Ji et al., 2026) also uses a DAG, but its paths couple token candidates within one denoising step rather than encoding reveal prerequisites. MAPLE instead uses offline, training-only graphs from attention or counterfactual ablation (Cohen-Wang et al., 2025; 2024).

6 EXPERIMENTS

Models and training. We evaluate MAPLE across two MDLM backbones from different model families and at different scales: LLaDA 2.0 16B MoE (Bie et al., 2025) and DreamReasoner-8B (Wu et al., 2026c). We perform continued pretraining on randomly sampled subsets of FineMath-4+ (Hugging Face TB Research, 2024) for mathematical reasoning and FineCode (Peng, 2025) for code generation. FineMath contains 2.31B/2.25B training tokens under the LLaDA 2/DreamReasoner tokenizers, while FineCode contains 2.74B/2.51B. We use Qwen3-8B-Base (Qwen Team, 2025) as the frozen autoregressive teacher for dependency extraction. Within each backbone–corpus group, all continued-pretraining runs start from the same released checkpoint and source corpus and use matched model- forward budgets on a one-pass schedule at diffusion block size 64. Graphs are extracted offline once and discarded after training. The selected MAPLE runs use counterfactual graphs for LLaDA 2/FineMath and DreamReasoner/FineCode, and attention graphs for the other two backbone–corpus groups; Appendix B.3 gives their extraction settings and training configurations.

Tasks and metrics. We evaluate mathematical reasoning on MATH500 (Lightman et al., 2023; Hendrycks et al., 2021), AIME24 (Jia, 2024), AIME25 (Zhang & Math-AI Team, 2025), OlympiadBench (He et al., 2024), and AMC12-2025 (Nguyen & Tulabandhula, 2026; Art of Problem Solving), and code generation on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), and LiveCodeBench Pro (Zheng et al., 2025). We additionally evaluate text-to-SQL generation on the Spider development set (Yu et al., 2018). We report task accuracy and parallelism: generated response tokens committed per model forward pass.

Baselines and decoding. Our primary baseline uses the same backbone trained with the standard MDLM objective, corresponding to the empty-graph any-order objective in Equation 1. Additional state-of-the-art training baselines include Efficient-DLM (Fu et al., 2026), TRIMS (Chen et al., 2026), and PUMA (Kim et al., 2026b). To test whether inference-time reordering alone accounts for the gains, we also apply UNCODE (Huang et al., 2026), KLASS (Kim et al., 2025b), and CCD (Chen et al., 2025a) to the DreamReasoner any-order FineMath checkpoint. Unless stated otherwise, evaluation uses greedy blockwise decoding with 64-token blocks, benchmark-specific generation caps, and confidence threshold $\tau = 0.95$. At each step, we reveal the highest- confidence remaining token and every other token whose confidence is at least τ; the inference-only controls use their respective ordering rules. All MAPLE results use the standard confidence-based decoder; the graph is never consulted at inference.

6.1 TASK ACCURACY

Table 1: Accuracy (%). Indented rows are only applied during decoding and, due to resource limitations, were not run on code. Bold marks the column best within each backbone.

		Mathematical reasoning					Code generation			
Model	Method	MATH 500	AIME 24+25	Olympiad Bench	AMC12 2025	Math avg.	Human Eval	MBPP	LCB Pro	Code avg.
LLaDA 2.0 16B MoE	Any-order (empty graph)	83.80	26.67	49.11	57.14	54.18	81.70	77.80	3.54	54.35
	Efficient-DLM	83.80	25.00	47.03	57.14	53.24	81.71	78.99	4.25	54.98
	TRIMS	84.80	28.33	47.18	**61.90**	55.55	84.15	77.04	4.11	55.10
	PUMA	85.00	25.00	49.41	**61.90**	55.33	81.71	79.38	4.11	55.07
	MAPLE (ours)	**86.60**	**33.33**	**51.34**	**61.90**	**58.29**	**84.80**	**81.70**	**4.39**	**56.96**
DreamReasoner 8B	Any-order	90.40	45.00	52.37	76.19	65.99	85.98	83.27	15.01	61.42
	+ UNCODE	89.40	45.00	52.23	76.19	65.70	—	—	—	—
	+ KLASS	90.20	41.67	52.52	76.19	65.14	—	—	—	—
	+ CCD	85.00	36.67	48.96	73.81	61.11	—	—	—	—
	Efficient-DLM	89.40	35.00	53.12	76.19	63.43	82.32	78.60	**15.72**	58.88
	TRIMS	90.40	46.67	**53.26**	76.19	66.63	81.10	82.49	13.31	58.97
	PUMA	90.60	43.33	52.52	78.57	66.26	82.32	**85.21**	15.01	60.85
	MAPLE (ours)	**91.20**	**50.00**	52.52	**83.33**	**69.26**	**89.63**	81.71	15.44	**62.26**

Table 2: Allocated GPU-hours for LLaDA 2 continued pretraining on 32 GPUs.

Corpus	Any-order	Efficient-DLM	TRIMS	PUMA	MAPLE
FineMath	407.4	410.9 (+0.9%)	407.2 (−0.03%)	408.7 (+0.3%)	424.3 (+4.2%)
FineCode	469.1	452.0 (−3.6%)	443.9 (−5.4%)	463.1 (−1.3%)	476.2 (+1.5%)

Table 1 shows broad gains across backbones and domains. MAPLE is best or tied-best in 11 of the 14 backbone–benchmark comparisons and has the highest math and code average for both backbones. On DreamReasoner/FineMath, where all seven baselines are available, MAPLE reaches 69.26%, 2.63 points above the strongest baseline (TRIMS). Some baselines win individual benchmarks, but those gains do not persist across tasks.

6.2 ACCURACY–PARALLELISM–COST TRADE-OFFS

Accuracy–cost tradeoff. Table 2 shows that MAPLE adds only 4.2%/1.5% GPU-hours on FineMath/FineCode. These single-run totals exclude queueing, preprocessing, graph or bucket construction, and evaluation; FineCode timing and accuracy use separate empty-graph replicates.

Accuracy–parallelism tradeoff. Holding each final checkpoint from Table 1 fixed, we vary only the decoding threshold in Figure 1.

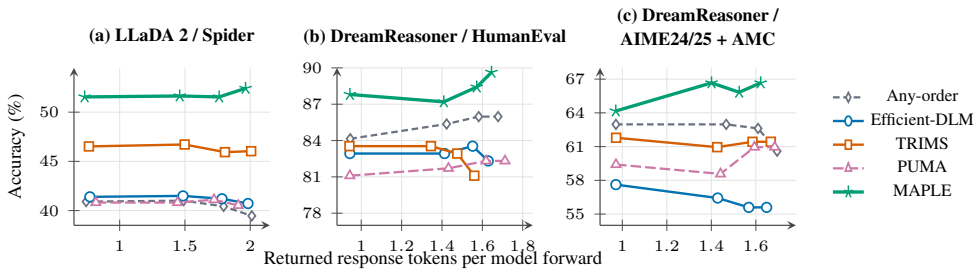


Figure 1: Accuracy–parallelism trade-offs from confidence-threshold sweeps of the same final checkpoints as Table 1 over $\tau \in \{0.95, 0.97, 0.99, 1.00\}$. The rightmost point of each curve is $\tau = 0.95$, the recommended setting for LLaDA 2 and DreamReasoner. Parallelism is returned response tokens per model forward.

At the shared $\tau = 0.95$ setting, MAPLE improves Spider normalized exact match from the strongest baseline's 46.03% to 52.42%, DreamReasoner HumanEval from 85.98% to 89.63%, and the DreamReasoner competition-math aggregate from 61.43% to 66.67%, while maintaining comparable returned-token parallelism. Thus the accuracy gains do not come from collapsing decoding toward the serial regime. A conservative response to prerequisite supervision would be to suppress confidence at most positions, leaving only a few above the threshold and reducing parallelism. The curves do not show this behavior. Instead, they are consistent with the position-selection KL redistributing confidence toward dependency-ready positions while preserving multiple simultaneous high-confidence predictions.

6.3 EXPLAINING THE GAINS: DEPENDENCY COMPLIANCE

For each method, we use the final LLaDA 2/FineMath checkpoint from Table 1, run the standard graph-free decoder, and record the completed response and the reveal round of every generated token. Only after generation do we extract dependencies from that response with each judge. For a same-block edge $j \to i$, we count a violation when j is committed in the same or a later round than i. This is therefore a free-running, generation-time test: it uses neither teacher forcing nor a ground-truth reveal trajectory.

Table 3: Dependency-violation rate (%, ↓) for LLaDA 2. Avg. is the unweighted dataset mean.

Method	MATH	A24	A25	AMC	OB	Avg.
Attention judge						
Any-order	8.46	8.20	7.93	8.11	8.76	8.29
Efficient-DLM	8.46	8.71	7.87	7.43	8.21	8.14
TRIMS	9.35	8.03	9.33	7.78	9.42	8.78
PUMA	8.64	9.23	8.79	8.68	8.85	8.84
MAPLE (ours)	**5.79**	**5.96**	**5.14**	**4.84**	**5.75**	**5.50**
Counterfactual judge						
Any-order	39.58	37.92	35.92	36.85	40.68	38.19
Efficient-DLM	41.31	39.50	36.96	35.86	40.57	38.84
TRIMS	39.82	37.35	35.81	38.03	41.36	38.47
PUMA	40.19	39.54	37.61	40.46	41.12	39.78
MAPLE (ours)	**26.05**	**23.87**	**24.17**	**24.41**	**24.49**	**24.60**
ValueFlow judge						
Any-order	44.56	**38.61**	47.16	46.92	52.11	45.87
Efficient-DLM	43.95	83.18	45.63	**25.34**	50.48	49.72
TRIMS	44.08	65.59	45.82	44.22	49.79	49.90
PUMA	47.24	43.83	43.14	41.72	56.31	46.45
MAPLE (ours)	**35.31**	41.56	**26.55**	41.42	**35.92**	**36.15**

The three judges use distinct signals: distance-debiased teacher attention (Appendix H.1), counterfactual target-probability drops after parent ablation (Appendix H.2), and teacher-free ValueFlow bindings and arithmetic dataflow (Appendix B.7). Because their densities differ, comparisons are within judge.

MAPLE beats any-order in 14 of 15 dataset–judge comparisons and ranks first in 13. It has the lowest average under all three judges, with reductions of 33.7%, 35.6%, and 21.2%. Transfer beyond its counterfactual training judge shows that MAPLE **elicits a general prerequisite-first generation behavior rather than fitting one extractor**.

Qualitative examples. Appendix I visualizes a running-sum calculation, an integrality check, and a shoelace-area computation. In all three, the any-order baseline commits a derived result before or alongside its supporting evidence and answers incorrectly; MAPLE reveals the relevant support first and answers correctly.

7 CONCLUSION

We introduced dependency-structured generation, which uses token-prerequisite DAGs to compactly characterize reveal behavior and unifies left-to-right autoregression, blockwise generation, and standard any-order MDLMs as boundary cases. MAPLE instantiates this view by extracting sequence-specific graphs offline from a frozen teacher and using them only to shape training; inference remains graph-free and retains the standard confidence-based parallel decoder. Its graph-induced objective is a tractable negative ELBO whose population global minimizers recover the data sequence distribution under transition realizability, without requiring the student to reproduce the teacher's reveal behavior. Across LLaDA 2 and DreamReasoner on mathematics and code, MAPLE improves accuracy without sacrificing parallelism and consistently reduces generation-time dependency violations across distinct judges. More broadly, MAPLE separates the source of order supervision from the deployed generation procedure: the teacher identifies useful prerequisite structure during training, while the student learns reveal behavior suited to its own parameterization. This separation makes dependency supervision modular: improved or domain-specific graph extractors can be incorporated without changing the student architecture or decoder.

Reproducibility Statement

Sections 2–4 define the graph teacher, training objectives, stochastic estimator, and graph-free decoder. Section 6 describes the primary comparisons and metrics. Appendix B records model and dataset revisions, corpus construction, optimization schedules, method-specific hyperparameters, random seeds, distributed training, decoding and scoring protocols, and compute accounting. Appendix H details dependency extraction and tokenizer projection. Within each backbone–corpus group, methods use the same initialization and source corpus and matched optimization-step and model- forward budgets. Upon acceptance, we will release the code, training recipes, and dependency artifacts.

AI use statement

We used generative AI tools to assist with code development and to revise the writing, and in a supporting role to give feedback on experiments and to help interpret results. We did not use them to design experiments, to develop the theoretical framework, mathematical claims, proofs or hypotheses, or to generate or annotate data. Generative models are also part of the method studied in this paper: we train diffusion language models and use a pretrained autoregressive model as a frozen teacher. The authors reviewed and tested all AI-assisted code and text, and take responsibility for the final content of this work.

References

Marianne Arriola and Volodymyr Kuleshov. Set diffusion: Interpolating token orderings between autoregression and diffusion for fast and flexible decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=tN8f0dxQ9i.

Marianne Arriola, Subham Sekhar Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin T Chiu, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Art of Problem Solving. AMC 12 problems and solutions. Online problem archive. URL https://artofproblemsolving.com/wiki/index.php?title=AMC_12_Problems_and_Solutions. Accessed 2026.

Hikaru Asano, Tadashi Kozuno, Kuniaki Saito, and Yukino Baba. Where-to-Unmask: Ground-Truth-Guided unmasking order learning for masked diffusion language models. *arXiv preprint arXiv:2602.09501*, 2026.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=h7-XixPCAL.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b. URL https://arxiv.org/abs/2108.07732.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

Kecheng Chen, Ziru Liu, Xijia Tao, Hui Liu, Xinyu Fu, Suiyun Zhang, Dandan Tu, Lingpeng Kong, Rui Liu, and Haoliang Li. Beyond confidence: Adaptive and coherent decoding for diffusion language models. *arXiv preprint arXiv:2512.02044*, 2025a. URL https://arxiv.org/abs/2512.02044.

Lingjie Chen, Ruizhong Qiu, Yuyu Fan, Yanjun Zhao, and Hanghang Tong. TRIMS: Trajectory-ranked instruction masked supervision for diffusion language models. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=7CN6O0qEVM.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dLLMs. *arXiv preprint arXiv:2509.26488*, 2025b.

CodeFuse and Ling Team. Every sample matters: Leveraging mixture-of-experts and high-quality data for efficient and accurate code LLM. *arXiv preprint arXiv:2503.17793*, 2025. URL https://arxiv.org/abs/2503.17793.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=7CMNSqsZJt.

Benjamin Cohen-Wang, Yung-Sung Chuang, and Aleksander Madry. Learning to attribute with attention. *arXiv preprint arXiv:2504.13752*, 2025.

Jia Deng, Junyi Li, Xin Zhao, Jinpeng Wang, Hongyu Lu, and Ji-Rong Wen. Beyond fully random masking: Attention-guided denoising and optimization for diffusion language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. URL https://aclanthology.org/2026.acl-long.2060/.

David Fox, Sam Bowyer, Song Liu, Laurence Aitchison, Raul Santos-Rodriguez, and Mengyue Yang. Learning generation orders for masked discrete diffusion models via variational inference. *arXiv preprint arXiv:2602.23968*, 2026.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: From autoregressive to diffusion language models, and beyond in speed. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ynRnOqI2W8.

Prateek Garg, Bhavya Kohli, and Sunita Sarawagi. Masked diffusion models are secretly learned-order autoregressive models. *arXiv preprint arXiv:2511.19152*, 2025. URL https://arxiv.org/abs/2511.19152.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j1tSLYKwg8.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2510.05725.

Chunsan Hong, Sanghyun Lee, and Jong Chul Ye. Unifying masked diffusion models with various generation orders and beyond. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2602.02112.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Lm8T39vLDTE.

Pengcheng Huang, Tianming Liu, Zhenghao Liu, Yukun Yan, Shuo Wang, Tong Xiao, Zulong Chen, and Maosong Sun. Empirical analysis of decoding biases in masked diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://arxiv.org/abs/2508.13021.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, et al. OpenCoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33167–33193, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1591. URL https://aclanthology.org/2025.acl-long.1591/.

Hugging Face TB Research. FineMath, 2024. URL https://huggingface.co/datasets/HuggingFaceTB/finemath.

Pengyu Ji, Zichen Zhang, Xiang Hu, and Kewei Tu. DA-DLM: Explicitly modeling token dependencies in diffusion language models. *arXiv preprint arXiv:2609.15070*, 2026. URL https://arxiv.org/abs/2609.15070.

Minghui Jia. AIME 2024 dataset. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024. Revision 8d88b2876a82a080e2f172cc9b25d0d9d2cb4792.

BumJun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. DAPD: Dependency-aware parallel decoding via attention for diffusion LLMs. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=LZro8iqjiA.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://arxiv.org/abs/2502.06768.

Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham Kakade, and Sitan Chen. Stop Training for the Worst: Progressive Unmasking Accelerates Masked Diffusion Training, 2026b. URL https://arxiv.org/abs/2602.10314.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=gOG9Zoyn4R.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Lizhuo Luo, Zhuoran Shi, Jiajun Luo, Zhi Wang, Shen Ren, Wenya Wang, and Tianwei Zhang. Dawn: Dependency-aware fast inference for diffusion llms, 2026. URL https://arxiv.org/abs/2602.06953.

Jama Hussein Mohamud, Mohsin Hasan, Mirco Ravanelli, and Yoshua Bengio. Adaptive order policies for masked diffusion. *arXiv preprint arXiv:2606.00295*, 2026. URL https://arxiv.org/abs/2606.00295.

Son The Nguyen and Theja Tulabandhula. MEMETRON: Memetic response optimizer for reward-guided post-decoding optimization of large language models. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=QRW8OGn3vb.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kpgURPRMGf.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

Fred Peng. Fine code, 2025. URL https://huggingface.co/datasets/fredzzp/fine_code. Dataset revision 53c4d84dd14183af0f80c6c11b34815b906f100b.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling, 2025. URL https://arxiv.org/abs/2502.03540.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Michael M. Bronstein, Anru Zhang, Alexander Tong, and Joey Bose. Planner aware path learning in diffusion language models training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lAlI5FuIf7.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-Guided parallel decoding in discrete diffusion language models. *arXiv preprint arXiv:2604.02560*, 2026.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=1NZ3DHF9nT.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Zirui Wu, Lin Zheng, Jiacheng Ye, Shansan Gong, Xueliang Zhao, Yansong Feng, Wei Bi, and Lingpeng Kong. DreamReasoner-8B: Block-size curriculum learning for diffusion reasoning models. *arXiv preprint arXiv:2606.19257*, 2026c. URL https://arxiv.org/abs/2606.19257.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/math-ai/aime25. Revision 563bb8404243c5f09de6ec262f2db674fe5bce9b.

Zihan Zheng, Zerui Cheng, Zeyu Shen, Shang Zhou, Kaiyuan Liu, Hansen He, Dongruixuan Li, Stanley Wei, et al. LiveCodeBench Pro: How do olympiad medalists judge LLMs in competitive programming? In *Advances in Neural Information Processing Systems*, volume 38, pp. 116099–116162, 2025. doi: 10.52202/085713-3502. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9712b78386cebdc3db7f1a48c2d20edb-Paper-Datasets_and_Benchmarks_Track.pdf.

APPENDIX CONTENTS

A Broader Diffusion Language Model Background **16**

B Implementation Details **16**

B.1 Models, Tokenization, and Corpora 16

B.2 Optimization and Training Budget 17

B.3 Dependency Extraction 17

B.4 Baselines 17

B.5 Decoding and Evaluation 18

B.6 Software Environments 18

B.7 Dependency-Compliance Evaluation 18

B.8 Compute Accounting 19

C Proof of Proposition 1 **19**

D Proof of Proposition 2 **19**

E Proof of Theorem 1 **21**

F A Generic Joint-to-Marginal Bound **23**

G Proof of Proposition 3 **23**

H Dependency Graph Extraction **24**

H.1 Attention-Based Extraction 24

H.2 Counterfactual Extraction 25

I Case Study: Visualizing the Decoding Process **25**

A Broader Diffusion Language Model Background

Discrete diffusion models such as D3PM (Austin et al., 2021a) corrupt discrete data and learn to reverse the corruption. Masked diffusion language models specialize this construction to an absorbing `[MASK]` state; MDLM (Sahoo et al., 2024) and MD4 (Shi et al., 2024) derive weighted masked-token objectives, while ARDM (Hoogeboom et al., 2022) and RADD (Ou et al., 2025) connect them to any-order autoregressive modeling. Large-scale instances include DiffuLLaMA (Gong et al., 2025), LLaDA (Nie et al., 2025), Dream (Ye et al., 2025), and LLaDA 2 (Bie et al., 2025). Block Diffusion (Arriola et al., 2025) and Fast-dLLM v2 (Wu et al., 2026a) combine autoregressive blocks with within-block diffusion, while Set Diffusion (Arriola & Kuleshov, 2026) extends blocks to arbitrary token sets. Generic confidence-based accelerators such as Fast-dLLM (Wu et al., 2026b) and KLASS (Kim et al., 2025b) remain applicable to MAPLE because its inference interface is unchanged.

B Implementation Details

This appendix specifies the configurations underlying Section 6. All reported training runs are full-parameter continued pretraining from the named public checkpoints. Unless explicitly stated otherwise, methods within a backbone–corpus group share the same initialization, source corpus, optimization-step budget, and evaluation protocol.

B.1 Models, Tokenization, and Corpora

We initialize LLaDA 2 from `inclusionAI/LLaDA2.0-mini` at revision `dad945cac317da394b390f82c7b40691d8a881ed`, and DreamReasoner from `Dream-org/DreamReasoner-8B` at revision `ed62b1d2c82ccd234b05ed2463b4c0ee640f2068`. The Dream packed releases use the tokenizer from `Dream-org/DreamReasoner-8B-Base` at revision `098c19582a619b634422caf129f7714177167cd8`. Dependency extraction uses the frozen teacher `Qwen/Qwen3-8B-Base` at revision `49e3418fbbbca6ecbdf9608b4d22e5a407081db4`.

The FineMath data come from the `finemath-4plus` configuration of `HuggingFaceTB/finemath@e92b25a616738fe95dc186b64dfb19f9c8525594`. We retain annotated subshards 0–7 of each 16-way parent shard and omit teacher-skipped overlong documents. FineCode comes from `fredzzp/fine_code@53c4d84dd14183af0f80c6c11b34815b906f100b` and uses a deterministic one-eighth subset selected by SHA-256 hashing, followed by the same graph-sidecar coverage filter. FineCode aggregates OpenCoder annealing data (Huang et al., 2025) and Ling-Coder-SyntheticQA (CodeFuse & Ling Team, 2025). All releases are packed to length 2,048, with document boundaries aligned to independent 64-token diffusion blocks. Valid-token and document-segment masks exclude padding and cross-document dependencies from the objective.

Table 4: Training-corpus sizes in tokens under each backbone's tokenizer. Counts exclude packing padding and dynamically admitted terminal-EOS targets; they are corpus sizes, not padded compute budgets.

Backbone tokenizer	Corpus	Training tokens
LLaDA 2	FineMath	2,311,964,569
LLaDA 2	FineCode	2,743,640,864
DreamReasoner	FineMath	2,247,259,426
DreamReasoner	FineCode	2,511,469,703

The corresponding packed-data manifest SHA-256 prefixes, in table order, are `e20f3ecd`, `45e250f9`, `1085241e`, and `4a0aece1`; the released manifests store the full digests.

B.2 Optimization and Training Budget

All runs use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, cosine learning-rate decay to one tenth of the peak, 5% linear warmup from zero, weight decay 0.1, gradient-norm clipping at 1.0, microbatch size 1 per device, gradient checkpointing, BF16 mixed precision, and training seed 42. Table 5 gives the remaining settings. Each schedule traverses 99.4–100.0% of its packed corpus.

Table 5: Selected continued-pretraining configurations. GBS is the global sequence batch size; hardware is shown for the selected MAPLE run.

Backbone	Corpus	Hardware	GBS	Updates	Peak $\rightarrow$ minimum LR
LLaDA 2	FineMath	32 B200	32	47,360	$5\times10^{-7} \rightarrow 5\times10^{-8}$
LLaDA 2	FineCode	32 B200	32	53,248	$5\times10^{-8} \rightarrow 5\times10^{-9}$
DreamReasoner	FineMath	64 B200	64	22,144	$5\times10^{-7} \rightarrow 5\times10^{-8}$
DreamReasoner	FineCode	64 GB300	64	24,256	$5\times10^{-7} \rightarrow 5\times10^{-8}$

Each method is evaluated at the checkpoint from its final training update; we do not select among intermediate checkpoints. All primary runs use one training seed; evaluation uses seed 17 and one generation per problem.

LLaDA 2 uses FSDP2 full sharding with expert parallelism 8, BF16 computation, FP32 reductions, and an FP32 MoE router. DreamReasoner uses FSDP full sharding with BF16 parameters and FP32 reductions. For DreamReasoner/FineCode, any-order and MAPLE use 64 GB300 GPUs; Efficient-DLM and PUMA use 64 B200 GPUs; and TRIMS uses 32 B200 GPUs with gradient accumulation 2, preserving GBS 64.

B.3 Dependency Extraction

We extract each graph once, before student training. The attention extractor averages the teacher predictor's attention over its final eight layers and all heads, removes the self-position and first-token sink, renormalizes, and divides each score by the mean at the same token distance. It retains scores of at least 5 and at most eight parents per token. The counterfactual extractor measures the distance-1 marginal effect in a 128-token teacher window, retains effects of at least 2 nats, and applies the same top-8 limit. Appendix H gives the algorithms and token-alignment procedure.

Graphs are discarded after training and are never used by the decoder.

Practical single-step estimator. For each independent mutable block of length L_s, we sample $K \sim \mathrm{Unif}\{0, \ldots, L_s - 1\}$, roll in one K-step prefix from $\tilde{h}^{\varepsilon}_{G}$, and evaluate all remaining token losses in one model forward pass. Multiplying the sampled bracket in Equation 15 by L_s is unbiased for its unnormalized sum over steps; weighting this sample by L_s and dividing by the number of mutable tokens in the batch estimates the corpus-token-normalized objective. The projected term introduces supervision between the ready and non-ready groups while retaining the current within-group ratios in its stop-gradient target. Were κ implemented as an independent head, additional supervision would be needed to determine those rankings; our implementation instead derives them from the token logits.

B.4 Baselines

The any-order baseline uses the matched one-state empty-graph estimator and the standard confidence decoder. Its expectation equals the standard any-order objective, although its mask sampler is not the literal corruption sampler. Efficient-DLM is our implementation of position-dependent masking with half-life ratio 0.1 because no official training code was available. TRIMS uses upstream revision `30d5f8c`, the same Qwen teacher, eight hard-to-easy NLL buckets, trajectory probability 0.5, future-mask probability 0.95, visible-random probability 0.05, exposure 0.8, loss weights 0.6/0.4, bucket ratio 0.6, and time clipping 10^{-3}. PUMA uses upstream revision `2f97f87`, eleven schedule points beginning at zero, confidence threshold 0.9, and the released pool/stall/resume policy. These training baselines use the same initialization and source corpus and matched optimizer- step and

model-forward budgets. Because PUMA chains several partial states from one example, it encounters fewer distinct corpus tokens under that fixed forward budget.

UNCODE, KLASS, and CCD are inference-only controls applied to the DreamReasoner/FineMath any-order checkpoint; they do not alter its training.

B.5 Decoding and Evaluation

Unless specified below, we use greedy blockwise confidence decoding with 64-token blocks and confidence threshold $\tau = 0.95$. Each step reveals the highest-confidence remaining token and every other token with confidence at least τ. We use evaluation seed 17 and one generation per problem. Mathematical reasoning uses a 4,096-token generation cap. LLaDA 2 uses 1,024 tokens for HumanEval and MBPP, whereas DreamReasoner uses 4,096; LiveCodeBench Pro uses 4,096 and Spider uses 256. We compute returned-token parallelism as the number of generated response tokens divided by the number of denoiser forward passes. The Pareto curves sweep $\tau \in \{0.95, 0.97, 0.99, 1.00\}$ while holding the checkpoint and all other settings fixed. DreamReasoner uses its released chat template with thinking enabled; LLaDA 2 uses its released native template. Exact prompts, stopping rules, and scoring scripts will accompany the released evaluation recipes.

Table 6: Evaluation sets and primary accuracy metrics. All metrics are single-sample pass@1.

Benchmark	Examples	Primary metric
MATH500 (Lightman et al., 2023; Hendrycks et al., 2021)	500	symbolic equivalence
AIME 2024/2025 (Jia, 2024; Zhang & Math-AI Team, 2025)	30/30	final integer answer
OlympiadBench (He et al., 2024)	674	reviewed symbolic answer
AMC12-2025 (Nguyen & Tulabandhula, 2026; Art of Problem Solving)	42	symbolic equivalence
HumanEval (Chen et al., 2021)	164	isolated execution pass@1
MBPP sanitized (Austin et al., 2021b)	257	isolated execution pass@1
LiveCodeBench Pro (Zheng et al., 2025)	706	testcase execution pass@1
Spider dev (Yu et al., 2018)	1,034	normalized exact-set match

LiveCodeBench Pro is evaluated with LightCPVerifier revision `021d121c882e70856b66113142495e31b9fe1d80` and runtime revision `d614eef4f95b3654bf0758c64e1194b2575850b8`. Spider uses the official database split and compares parsed SQL structures after deterministic normalization; we retain the unmodified upstream exact-match result separately. This normalized score is our deterministic extension of the upstream structural evaluator; we do not report execution accuracy. The math macro first pools AIME24 and AIME25 by example count, then averages MATH500, pooled AIME, OlympiadBench, and AMC12-2025 equally. The code macro is the unweighted mean of HumanEval, MBPP, and LiveCodeBench Pro. Returned-token parallelism excludes EOS and discarded terminal-block or alignment positions.

B.6 Software Environments

The B200 runs use Python 3.11.13, PyTorch 2.7.1 with CUDA 12.8, and NCCL 2.26.2. LLaDA 2 training uses Transformers 4.56.2 and FlashAttention 2.7.4.post1; its evaluation uses Transformers 4.57.1. DreamReasoner B200 runs use Transformers 5.7.0. DreamReasoner/FineCode runs on GB300 use Python 3.12.9, PyTorch 2.12.0a0, CUDA 13.1, NCCL 2.30.100, and FlashAttention 2.7.4.post1.

B.7 Dependency-Compliance Evaluation

We apply all three judges to the same stored generations. An edge is countable only when both endpoints are generated response tokens in the same actual 64-token decoder block. For an edge $j \to i$, a violation occurs when the commit round of j is equal to or later than that of i; simultaneous commits therefore count as violations. Dataset rates pool countable edges across examples, and the reported macro is the unweighted mean of dataset rates.

The attention and counterfactual judges use the Qwen teacher and the extraction procedures in Appendix H.

ValueFlow judge. ValueFlow (`valueflow_stage_a@4`) is a deterministic, teacher-free parser that builds a dependency graph directly from each completed response. It emits relations for uniquely resolved bindings and their uses, exact computations, value reuse, aggregations, and final-answer derivations, while abstaining when provenance is ambiguous. Its 34-parent capacity is storage capacity rather than top-k pruning; the reported run had no truncated or overflow-rejected relations. Exact character spans are projected to generated token coordinates using the pinned LLaDA 2 tokenizer. Examples that fail exact prompt-and-response round-trip alignment are excluded and counted separately.

B.8 Compute Accounting

Table 2 reports allocated training GPU-hours, computed as the outer Slurm allocation's elapsed wall time times the number of allocated GPUs. Each cell is one timing run. The totals include the training allocation but exclude queueing, corpus preprocessing, dependency-graph or TRIMS-bucket construction, and evaluation. FineCode timing and accuracy use separate replicates of the same effective configuration. We do not report DreamReasoner training time because a complete matched timing grid is unavailable.

C Proof of Proposition 1

Fix a completed sequence x. For any step k and prefix O_k, expanding the local action KL and combining it with the token term gives

$$\begin{aligned}&\mathrm{KL}(h_G(\cdot \mid O_k, x) \,\|\, \kappa_\theta(\cdot \mid z_k)) + \mathbb{E}_{I_k \sim h_G}\left[-\log \pi_{\theta, I_k}(x_{I_k} \mid z_k)\right] \\ &\quad = \mathbb{E}_{I_k \sim h_G}\left[\log h_G(I_k \mid O_k, x) - \log T_\theta(I_k, x_{I_k} \mid z_k)\right]. \end{aligned} \tag{16}$$

Here we used $T_\theta(i, x_i \mid z_k) = \kappa_\theta(i \mid z_k)\pi_{\theta,i}(x_i \mid z_k)$.

At every proper prefix, $h_G(\cdot \mid O_k, x)$ is normalized over $U(O_k)$. Iteratively summing over $i_{L-1}, \ldots, i_0$ therefore gives $\sum_\gamma q_G(\gamma \mid x) = 1$; a transition assigned zero probability by h_G contributes zero.

For a draw $\gamma = (I_0, \ldots, I_{L-1}) \sim q_G(\cdot \mid x)$, d_k^G is the distribution of O_k. For every O_k in its support, the next position is distributed as $h_G(\cdot \mid O_k, x)$. Hence, for every function f_k,

$$\mathbb{E}_{O_k \sim d_k^G} \mathbb{E}_{I_k \sim h_G(\cdot \mid O_k, x)}[f_k(O_k, I_k)] = \mathbb{E}_{\gamma \sim q_G(\cdot \mid x)}[f_k(O_k, I_k)]. \tag{17}$$

Applying Equation 17 to Equation 16, summing over k, and using the product definitions of q_G and p_θ yields

$$\begin{aligned}\mathcal{L}_G(x; \theta) &= \mathbb{E}_{\gamma \sim q_G(\cdot \mid x)}\left[\log \prod_{k=0}^{L-1} h_G(I_k \mid O_k, x) - \log \prod_{k=0}^{L-1} T_\theta(I_k, x_{I_k} \mid Z_k)\right] \\ &= \mathbb{E}_{\gamma \sim q_G(\cdot \mid x)}\left[\log q_G(\gamma \mid x) - \log p_\theta(x, \gamma)\right]. \end{aligned} \tag{18}$$

When $p_\theta(x) > 0$, factor $p_\theta(x, \gamma) = p_\theta(x) p_\theta(\gamma \mid x)$ to obtain

$$\begin{aligned}\mathcal{L}_G(x; \theta) &= -\log p_\theta(x) + \mathbb{E}_{\gamma \sim q_G(\cdot \mid x)} \log \frac{q_G(\gamma \mid x)}{p_\theta(\gamma \mid x)} \\ &= -\log p_\theta(x) + \mathrm{KL}(q_G(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)). \end{aligned} \tag{19}$$

Nonnegativity of KL gives Equation 10. If $p_\theta(x) = 0$, every joint path probability is zero, so $\mathcal{L}_G(x; \theta) = +\infty$ and the lower bound still holds. If $p_\theta(x) > 0$ but $p_\theta(x, \gamma) = 0$ on a path with positive $q_G(\gamma \mid x)$, the posterior is defined and both the path KL and the loss are $+\infty$. Thus all stated inequalities remain valid in the extended-real sense. □

D Proof of Proposition 2

Under the proposition's hypotheses $h_G = \bar{h}_G$ and $\kappa_\theta = \bar{h}_G$, every local action KL in Equation 9 is zero. It therefore remains only to identify the graph-induced occupancies and token terms.

Autoregressive chain. For the chain graph, every proper prefix state has the singleton ready set $R_G(\{1,\dots,k\}) = \{k+1\}$. Thus q_G puts unit mass on the left-to-right order. Equation 9 becomes

$$\mathcal{L}_G(x;\theta) = \sum_{i=1}^{L} -\log \pi_{\theta,i}\left(x_i \mid z(\{1,\dots,i-1\}, x)\right), \tag{20}$$

the left-to-right NLL conditional on (c, L). Since the only supported path is left-to-right, marginalizing the path also gives $p_\theta(x) = \prod_{i=1}^{L} \pi_{\theta,i}(x_i \mid z(\{1,\dots,i-1\}, x))$.

Empty graph and the standard any-order MDLM population loss. For the empty graph, $R_G(O) = \mathcal{I} \setminus O$, so every order has probability $\prod_{k=0}^{L-1}(L-k)^{-1} = 1/L!$. Let M denote the masked set, let $m = |M|$, and abbreviate the token loss at that state by

$$\ell_i(M) := -\log \pi_{\theta,i}\left(x_i \mid z(\mathcal{I} \setminus M, x)\right). \tag{21}$$

At step $k = L - m$ of a uniform permutation, the remaining set M is uniform over the $\binom{L}{m}$ subsets of size m, and the next position is uniform within M. Therefore the contribution of that step is $\binom{L}{m}^{-1} \sum_{|M|=m} m^{-1} \sum_{i\in M} \ell_i(M)$. Summing over $m = 1,\dots,L$ gives

$$\mathcal{L}_{\mathrm{AO}} = \sum_{m=1}^{L} \frac{1}{\binom{L}{m}} \sum_{\substack{M\subseteq\mathcal{I}\\|M|=m}} \frac{1}{m} \sum_{i\in M} \ell_i(M). \tag{22}$$

Now sample a mask rate $t \sim \mathrm{Unif}(0,1)$ and mask each position independently with probability t. The corresponding inverse-rate-weighted population token loss expands as

$$\begin{aligned} &\mathbb{E}_{t\sim\mathrm{Unif}(0,1)}\mathbb{E}_{M\sim\mathrm{Bern}(t)^L}\left[\frac{1}{t}\sum_{i\in M}\ell_i(M)\right] \\ &\quad= \sum_{\varnothing\neq M\subseteq\mathcal{I}} \left(\int_0^1 t^{|M|-1}(1-t)^{L-|M|}\,dt\right) \sum_{i\in M}\ell_i(M). \end{aligned} \tag{23}$$

For a set of size m, the beta integral in parentheses is

$$\int_0^1 t^{m-1}(1-t)^{L-m}\,dt = \frac{(m-1)!(L-m)!}{L!} = \frac{1}{m\binom{L}{m}}. \tag{24}$$

Substitution into Equation 23 reproduces Equation 22 term by term. Thus

$$\mathcal{L}_G = \mathcal{L}_{\mathrm{AO}} = \mathbb{E}_{t\sim\mathrm{Unif}(0,1)}\mathbb{E}_{M\sim\mathrm{Bern}(t)^L}\left[\frac{1}{t}\sum_{i\in M} -\log \pi_{\theta,i}(x_i \mid z(\mathcal{I}\setminus M, x))\right]. \tag{25}$$

The empty mask contributes zero (with the value at $t = 0$ irrelevant because it is a measure-zero event). Under unclipped uniform mask-rate sampling, independent masking, inverse-rate weighting, and no additional raw-rate input, Equation 25 is the summed standard any-order MDLM population token loss. Dividing both sides by L gives its mean-per-token convention.

Ordered blocks. Let $\mathcal{I} = B_1 \dot\cup \cdots \dot\cup B_J$. Place an edge from every position in B_b to every position in B_{b+1}, and place no edges within a block. By transitive reachability, every position in an earlier block is an ancestor of every position in a later block. Hence all of $B_{<b} := \bigcup_{a<b} B_a$ must be revealed before any position in B_b is ready, while every unrevealed position of the active block is ready. Consequently, the reveal-order distribution completes the blocks in their fixed order and chooses a uniform permutation independently within each block. Writing $\mathfrak{S}(B_b)$ for the permutations of B_b, this gives

$$\mathcal{L}_G(x;\theta) = \sum_{b=1}^{J} \mathbb{E}_{\sigma_b\sim\mathrm{Unif}(\mathfrak{S}(B_b))} \sum_{r=0}^{|B_b|-1} -\log \pi_{\theta,\sigma_{b,r}}\left(x_{\sigma_{b,r}} \mid z_{b,r}\right), \tag{26}$$

where $z_{b,r}$ reveals $B_{<b}$ and the first r positions of σ_b. For each block, the same subset argument as Equation 22 converts its uniform-permutation expectation to uniform random-mask token training within that block. Since later blocks cannot be selected early, summing the model joint over the within-block orders also yields the block-autoregressive factorization

$$p_\theta(x) = \prod_{b=1}^{J} \left[\frac{1}{|B_b|!} \sum_{\sigma_b \in \mathfrak{S}(B_b)} \prod_{r=0}^{|B_b|-1} \pi_{\theta,\sigma_{b,r}}(x_{\sigma_{b,r}} \mid z_{b,r}) \right]. \tag{27}$$

The chain and ordered-block constructions also yield the displayed sequential generative factorizations. A bidirectional masked Transformer still defines these conditionals because future masks reveal no token values. Literal recovery of conventional autoregressive and blockwise implementations additionally requires their causal or block-causal attention and input conventions. The any-order token-objective identity does not by itself identify the sequential generator with a particular parallel diffusion process. □

E Proof of Theorem 1

Work conditionally on the fixed context and length (c, L). Let μ_0 denote the common distribution of the initial all-mask state Z_0 (a point mass in our setting), and let $a = (i, v)$ denote a position–token event. Write $F(z, a)$ for the deterministic update that places token v at masked position i. Under $\rho_G(x, \gamma) = p_{\mathrm{data}}(x) q_G(\gamma \mid x)$, define $Y_k = (I_k, X_{I_k})$. For each k and every state z with $\rho_G(Z_k = z) > 0$, define

$$\overline{T}_G(a \mid z) := \rho_G(Y_k = a \mid Z_k = z). \tag{28}$$

Because Z_k has exactly k revealed modeled positions, the supports of Z_k at different steps are disjoint, so this defines one unambiguous state-conditioned kernel. Outside this support, define $\overline{T}_G$ as any normalized distribution over valid actions.

Using Equation 16 and averaging over $X \sim p_{\mathrm{data}}$ gives

$$\mathcal{J}_G(\theta) = A_G + \sum_{k=0}^{L-1} \mathbb{E}_{\rho_G} \left[-\log T_\theta(Y_k \mid Z_k) \right], \tag{29}$$

where

$$A_G := \sum_{k=0}^{L-1} \mathbb{E}_{\rho_G} \left[\log h_G(I_k \mid O_k, X) \right] \tag{30}$$

does not depend on θ.

For each step, the tower property and the cross-entropy identity give

$$\begin{aligned} &\mathbb{E}_{\rho_G}[-\log T_\theta(Y_k \mid Z_k)] \\ &\quad = \mathbb{E}_{Z_k \sim \rho_G} \sum_a \overline{T}_G(a \mid Z_k) \left[-\log T_\theta(a \mid Z_k) \right] \\ &\quad = \mathbb{E}_{Z_k \sim \rho_G} \left[H\big(\overline{T}_G(\cdot \mid Z_k)\big) + \mathrm{KL}\big(\overline{T}_G(\cdot \mid Z_k) \,\|\, T_\theta(\cdot \mid Z_k)\big) \right]. \end{aligned} \tag{31}$$

Here $H(r) := -\sum_a r(a) \log r(a)$. Summing this identity gives

$$\mathcal{J}_G(\theta) = C_G + \sum_{k=0}^{L-1} \mathbb{E}_{Z_k \sim \rho_G} \mathrm{KL}\big(\overline{T}_G(\cdot \mid Z_k) \,\|\, T_\theta(\cdot \mid Z_k)\big), \tag{32}$$

where

$$C_G := A_G + \sum_{k=0}^{L-1} \mathbb{E}_{Z_k \sim \rho_G} H\big(\overline{T}_G(\cdot \mid Z_k)\big) \tag{33}$$

is independent of θ. All KL terms in Equation 32 are nonnegative. Transition realizability supplies one common parameter setting that makes every term zero, so the lower bound is attained and $\mathcal{J}_G^\star = C_G$.

To see why independent universal position-selection and token-prediction heads suffice for this assumption, factor each normalized target transition as

$$\kappa^{\mathrm{tar}}(i \mid z) := \sum_{v \in \mathcal{V}} \overline{T}_G((i, v) \mid z), \tag{34}$$

$$\pi_i^{\mathrm{tar}}(v \mid z) := \frac{\overline{T}_G((i, v) \mid z)}{\kappa^{\mathrm{tar}}(i \mid z)} \quad \text{when } \kappa^{\mathrm{tar}}(i \mid z) > 0. \tag{35}$$

The dependence of these target functions on the fixed reference ρ_G is suppressed in their notation; neither takes G as an input. When $\kappa^{\mathrm{tar}}(i \mid z) = 0$, define the associated token conditional arbitrarily. The product $\kappa^{\mathrm{tar}}(i \mid z)\pi_i^{\mathrm{tar}}(v \mid z)$ then equals $\overline{T}_G((i, v) \mid z)$ everywhere. Since the state supports at different steps are disjoint, a single pair of state-conditioned heads can represent all of these factors as functions of z alone. Idealized universal categorical heads whose model class includes the simplex boundary can represent these factors simultaneously. Ordinary finite-logit softmax heads can only approach exact zeros in the limit; this is why Theorem 1 states transition realizability explicitly.

On a complete trajectory $\tau = (z_0, a_0, z_1, \ldots, a_{L-1}, z_L)$, define the state-Markovized reference and model trajectory distributions by

$$\bar{\rho}_G(\tau) := \mu_0(z_0) \prod_{k=0}^{L-1} \left[\overline{T}_G(a_k \mid z_k)\mathbf{1}\{z_{k+1} = F(z_k, a_k)\}\right], \tag{36}$$

$$p_\theta^{\mathrm{tr}}(\tau) := \mu_0(z_0) \prod_{k=0}^{L-1} \left[T_\theta(a_k \mid z_k)\mathbf{1}\{z_{k+1} = F(z_k, a_k)\}\right]. \tag{37}$$

Because each transition fills one previously masked position, these are equivalently distributions over a completed sequence and its reveal order. The model trajectory distribution p_θ^{tr} is the trajectory representation of the joint $p_\theta(x, \gamma)$ from Equation 3. The two trajectory distributions share the initial distribution and deterministic update, but $\bar{\rho}_G$ need not equal the trajectory representation of the original, generally non-Markov joint ρ_G.

We next prove by induction that $\bar{\rho}_G$ and ρ_G have the same state marginal at every step. The claim holds at $k = 0$ because both use μ_0. If it holds at step k, then for every next state z',

$$\begin{aligned}
\bar{\rho}_G(Z_{k+1} = z') &= \sum_{z,a} \bar{\rho}_G(Z_k = z)\overline{T}_G(a \mid z)\mathbf{1}\{F(z, a) = z'\} \\
&= \sum_{z,a} \rho_G(Z_k = z)\rho_G(Y_k = a \mid Z_k = z)\mathbf{1}\{F(z, a) = z'\} \\
&= \rho_G(Z_{k+1} = z').
\end{aligned} \tag{38}$$

The last equality uses $Z_{k+1} = F(Z_k, Y_k)$ under ρ_G. The induction proves $\bar{\rho}_G(Z_k) = \rho_G(Z_k)$ for every k. Since Z_L is the fully revealed sequence and

$$\begin{aligned}
\sum_\gamma \rho_G(x, \gamma) = p_{\mathrm{data}}(x) \sum_\gamma q_G(\gamma \mid x) = p_{\mathrm{data}}(x), \\
\bar{\rho}_G(X = x) = \rho_G(X = x) = p_{\mathrm{data}}(x).
\end{aligned} \tag{39}$$

Applying the KL chain rule to Equations 36–37, the identical deterministic update factors cancel and give

$$\begin{aligned}
\mathrm{KL}(\bar{\rho}_G \| p_\theta^{\mathrm{tr}}) &= \mathrm{KL}(\mu_0 \| \mu_0) + \sum_{k=0}^{L-1} \mathbb{E}_{Z_k \sim \bar{\rho}_G} \mathrm{KL}\big(\overline{T}_G(\cdot \mid Z_k) \,\|\, T_\theta(\cdot \mid Z_k)\big) \\
&= \mathcal{J}_G(\theta) - C_G = \mathcal{J}_G(\theta) - \mathcal{J}_G^\star.
\end{aligned} \tag{40}$$

The initial-state KL is zero. The second equality uses the state-marginal identity proved above together with Equation 32.

Finally, map each trajectory to its completed sequence. Equation 39 shows that this maps $\bar{\rho}_G$ to p_{data}, while Equation 3 shows that it maps p_θ^{tr} to p_θ. KL data processing therefore gives

$$\mathrm{KL}(p_{\mathrm{data}} \| p_\theta) \le \mathrm{KL}(\bar{\rho}_G \| p_\theta^{\mathrm{tr}}) = \mathcal{J}_G(\theta) - \mathcal{J}_G^\star, \tag{41}$$

which proves Equation 13. At any global minimizer, the right-hand side is zero, so $\mathrm{KL}(p_{\mathrm{data}} \| p_{\theta^\star}) = 0$ and $p_{\theta^\star} = p_{\mathrm{data}}$.

All cross-entropies and KL divergences above use the extended-real convention. If the model assigns zero probability to a transition with positive $\overline{T}_G$ probability, the corresponding conditional KL, trajectory KL, and objective are $+\infty$, so the identities and inequalities remain valid. The finite-horizon terms in C_G are finite. □

F A Generic Joint-to-Marginal Bound

Proposition 4 (Joint fitting controls sequence fitting)**.** *For every θ,*

$$\mathrm{KL}(p_{\mathrm{data}}(x) \,\|\, p_\theta(x)) \le \mathrm{KL}(\rho_G(x,\gamma) \,\|\, p_\theta(x,\gamma)) = \mathcal{J}_G(\theta) - H(p_{\mathrm{data}}). \tag{42}$$

The left KL is over completed sequences x, whereas the right KL is over the joint space (x, γ). Recall that $\rho_G(x,\gamma) = p_{\mathrm{data}}(x) q_G(\gamma \mid x)$ and define $H(p_{\mathrm{data}}) := -\mathbb{E}_{X \sim p_{\mathrm{data}}} \log p_{\mathrm{data}}(X)$. Direct expansion gives

$$\begin{aligned} \mathrm{KL}(\rho_G(x,\gamma) \,\|\, p_\theta(x,\gamma)) &= \mathbb{E}_{(X,\gamma)\sim\rho_G} \left[\log p_{\mathrm{data}}(X) + \log q_G(\gamma \mid X) - \log p_\theta(X,\gamma)\right] \\ &= -H(p_{\mathrm{data}}) + \mathcal{J}_G(\theta), \end{aligned} \tag{43}$$

which proves the equality in Equation 42. The deterministic projection $(X, \gamma) \mapsto X$ maps the joint ρ_G to p_{data} because $\sum_\gamma q_G(\gamma \mid x) = 1$, and maps the joint model distribution $p_\theta(x,\gamma)$ to its marginal $p_\theta(x)$ by Equation 3. Applying KL data processing to this map proves the inequality. The statement remains valid in the extended-real sense when the joint ρ_G is not absolutely continuous with respect to the joint model distribution $p_\theta(x,\gamma)$. □

G Proof of Proposition 3

Soft-projection calculation. Fix a state and abbreviate its ready and blocked sets by R and $B = U \setminus R$. Let $S := \sum_{i \in R} \bar{\kappa}_{\theta,i}$ and $Z_\varepsilon := S + \varepsilon(1 - S)$. Then Equation 14 is

$$h^{\mathrm{proj},\varepsilon}_{\theta,G}(i) = \begin{cases} \bar{\kappa}_{\theta,i}/Z_\varepsilon, & i \in R, \\ \varepsilon \bar{\kappa}_{\theta,i}/Z_\varepsilon, & i \in B. \end{cases} \tag{44}$$

For $0 < \varepsilon \le 1$, set $\beta = -\log \varepsilon$. For any distribution q on U, direct substitution gives

$$\mathrm{KL}\Big(q \,\|\, h^{\mathrm{proj},\varepsilon}_{\theta,G}\Big) = \mathrm{KL}(q \,\|\, \bar{\kappa}_\theta) + \beta q(B) + \log Z_\varepsilon. \tag{45}$$

Thus $h^{\mathrm{proj},\varepsilon}_{\theta,G}$ uniquely minimizes $\mathrm{KL}(q\|\bar{\kappa}_\theta) + \beta q(B)$: it is a reverse-KL self-projection with a soft penalty for blocked mass. The limit $\varepsilon \downarrow 0$ restricts q to R and recovers the hard-support projection.

At the construction point, Equation 44 also gives

$$D_\varepsilon := \mathrm{KL}\Big(h^{\mathrm{proj},\varepsilon}_{\theta,G} \,\|\, \kappa_\theta\Big) = -\log Z_\varepsilon + \frac{\varepsilon(1-S)}{Z_\varepsilon} \log \varepsilon, \tag{46}$$

with the continuous convention $0 \log 0 = 0$. Hence $D_0 = -\log S$, $D_1 = 0$, and, for $0 < \varepsilon \le 1$,

$$\frac{dD_\varepsilon}{d\varepsilon} = \frac{S(1-S)}{Z_\varepsilon^2} \log \varepsilon \le 0. \tag{47}$$

The action penalty therefore decreases monotonically to zero as the graph constraint is softened.

The two fallback cases. If G is empty, $R_G(O) = U(O)$ for every proper O, so every gate in Equation 14 equals one for every ε. For the second case, take any nonempty G and set $\varepsilon = 1$; every gate again equals one. In either case, $h^{\mathrm{proj},\varepsilon}_{\theta,G} = \bar{\kappa}_\theta$. Because $\bar{\kappa}_\theta = \mathrm{sg}[\kappa_\theta]$ is numerically equal to κ_θ at the current construction point,

$$\mathrm{KL}(\bar{\kappa}_\theta \| \kappa_\theta) = 0. \tag{48}$$

The first argument is held fixed by stop-gradient, so differentiation at that same point gives

$$\begin{aligned}\nabla_\theta \mathrm{KL}(\bar{\kappa}_\theta \| \kappa_\theta) &= -\sum_i \frac{\bar{\kappa}_{\theta,i}}{\kappa_{\theta,i}} \nabla_\theta \kappa_{\theta,i} \\ &= -\nabla_\theta \sum_i \kappa_{\theta,i} = 0. \end{aligned} \tag{49}$$

Thus both the action-loss value and its gradient are zero.

For an empty graph, $\bar{h}_G = h_{\mathrm{AO}}(\cdot \mid O)$ and therefore $\tilde{h}^{\varepsilon}_G = h_{\mathrm{AO}}(\cdot \mid O)$ for every ε. For any nonempty G, setting $\varepsilon = 1$ similarly gives $\tilde{h}^{1}_G = h_{\mathrm{AO}}(\cdot \mid O)$. In either case sequential sampling assigns every reveal order probability $\prod_{k=0}^{L-1}(L-k)^{-1} = 1/L!$, so the revealed-set distribution and token term equal $\mathcal{L}_{\mathrm{AO}}$. Together with the zero action term, this proves the loss and training-gradient identities in Proposition 3 for both fallback cases.

Equation 1 identifies the resulting token loss with the standard any-order MDLM population objective. At inference, MAPLE evaluates neither G nor the projected target; applying the same confidence score and parallel commit schedule to a fixed token model π_θ gives exactly the vanilla decoding procedure. This decoder equality is algorithmic and does not identify the parallel kernel with the sequential likelihood model.

Contrast with the exact ELBO using $\tilde{h}^{\varepsilon}_G$. Let d_k^{AO} be the step-k revealed-set distribution induced by a uniform permutation. If the softened uniform-ready policy $\tilde{h}^{\varepsilon}_G$ is used consistently in the exact ELBO, then at $\varepsilon = 1$,

$$\mathcal{L}^{\mathrm{ELBO}}_{1,G} = \mathcal{L}_{\mathrm{AO}} + \sum_{k=0}^{L-1} \mathbb{E}_{O_k \sim d_k^{\mathrm{AO}}} \mathrm{KL}(h_{\mathrm{AO}}(\cdot \mid O_k) \,\|\, \kappa_\theta(\cdot \mid z_k)) \,. \tag{50}$$

It equals $\mathcal{L}_{\mathrm{AO}}$ if and only if $\kappa_\theta(\cdot \mid z_k) = h_{\mathrm{AO}}(\cdot \mid O_k)$ at every d_k^{AO}-supported state. The soft-projected hybrid instead replaces this uniform action target by the stop-gradient self-target at $\varepsilon = 1$, making its action KL zero for any κ_θ; that replacement is why the hybrid is not this exact ELBO. With fixed uniform κ, the exact-ELBO theorem applies to the resulting uniform sequential rollout subject to the assumptions of Theorem 1; it does not by itself characterize an external parallel confidence decoder. □

H DEPENDENCY GRAPH EXTRACTION

We extract a dependency graph for each training document with a frozen autoregressive teacher (Qwen3-8B-Base), using one of two scores for a candidate edge $j \to i$ from an earlier position j to a later position i. Restricting candidates to earlier-to-later edges guarantees acyclicity and matches the context available to the causal teacher; it is an extraction choice, while the framework itself permits arbitrary directed acyclic graphs. Because the autoregressive teacher and the diffusion student use different tokenizers, both scores are computed on teacher tokens and then projected onto the student's tokens through the overlap of their character spans; markup that covers no text neither gives nor receives edges.

H.1 ATTENTION-BASED EXTRACTION

A causal teacher predicts x_i at position $i-1$, so we read the attention row of this prediction query, averaged over heads and the last eight layers, and drop the entry of the query's own position, which is often the largest in its row. After removing the attention sink (the first token of the input) and renormalizing the row, the attention weight on an earlier token decays steeply with distance (Figure 2a), so ranking by the raw weight returns an almost left-to-right chain (Figure 2b). We therefore divide each weight by the average weight at the same distance within the document and keep j as a parent of i when this ratio is at least 5.

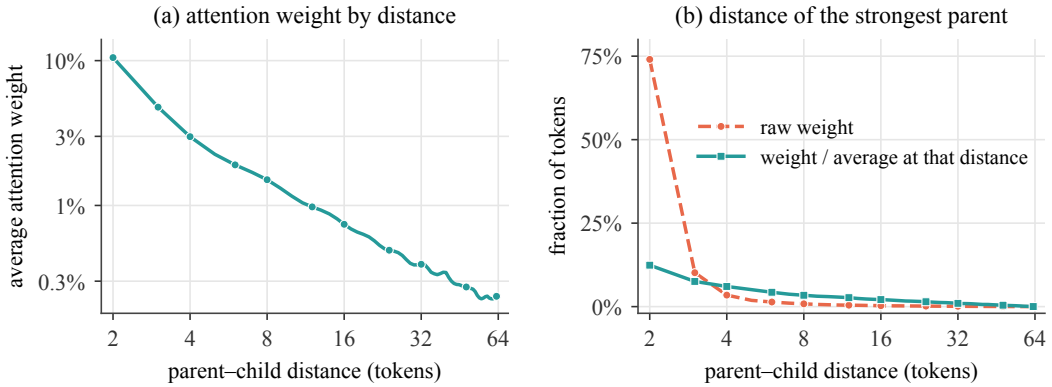


Figure 2: Attention of Qwen3-8B-Base on 1,400 FineMath-4+ documents . (a) Average attention weight on an earlier token j by parent–child distance $i - j$. (b) Distance of each token's strongest parent when parents are ranked by the raw weight or by the weight divided by the average weight at that distance.

H.2 Counterfactual Extraction

As in attention-mask ablation (Cohen-Wang et al., 2025), the effect of j on i is the drop in the teacher's log-probability of x_i when x_j is ablated by masking its key in every layer and head. This fails at distance $i - j = 1$: x_i is predicted at position j itself, so the prediction still sees x_j through that position's own hidden state, and the measured effect is close to zero. For $i = j + 1$, we instead replace x_j by the teacher's own predictions for it, averaging $p(x_i \mid x_{<j}, x')$ over its most likely tokens x'. We keep j as a parent of i when the effect is at least 2 nats, i.e., when ablating x_j makes x_i at least $e^2 \approx 7.4$ times less likely.

I Case Study: Visualizing the Decoding Process

We examine how teacher-derived dependency supervision changes the order in which MAPLE commits tokens, comparing it with the any-order model (empty graph) on the same problems. We use the LLaDA 2.0 16B MoE checkpoints from Table 1 and decode greedily with the confidence-based block decoder, with block size 64 and at most 4096 generated tokens, recording the decoding step at which each token is committed. The any-order model can commit a token before the tokens it depends on are revealed. In a math solution, this can mean that the result of a calculation is committed while the expression that produces it is still partly masked; in the selected examples below, this leads to a wrong final answer. Under the same decoder, MAPLE commits the corresponding results mostly after the tokens they depend on are revealed, and its final answers are correct. Figures 3–5 show one problem each from MATH500, AMC12-2025, and OlympiadBench, attempted by both models.

Each figure shows a short excerpt of both solutions after selected decoding steps (left). The decoder completes one 64-token block before it starts the next, and steps are counted within the block being decoded (next block: a step of the following block). Rows need not be consecutive steps, and the step numbers only order the rows of one model; they do not compare decoding speed between the two models. Each box is one token, shown as generated except that a token's leading space is not drawn (␣ marks a token that consists only of a space, and `\n` a line break). Gray boxes are still masked, and the shade of a committed box gives the step at which it was committed, from light (earlier) to dark (later). The tokens that decide the final answer have a solid border (red for the any-order model, green for MAPLE), and the tokens they are computed from have a dashed orange border. Vertical dots stand for the remaining steps, and the last row (Last step) shows the final answer once decoding has finished (× wrong, ✓ correct).

MATH500 (Figure 3). Both models add the digit sums $1 + 3 + 6 + \cdots + 36$ one term at a time. The any-order model commits the first digit of $35 + 21$ in the same step as the two operands and gets

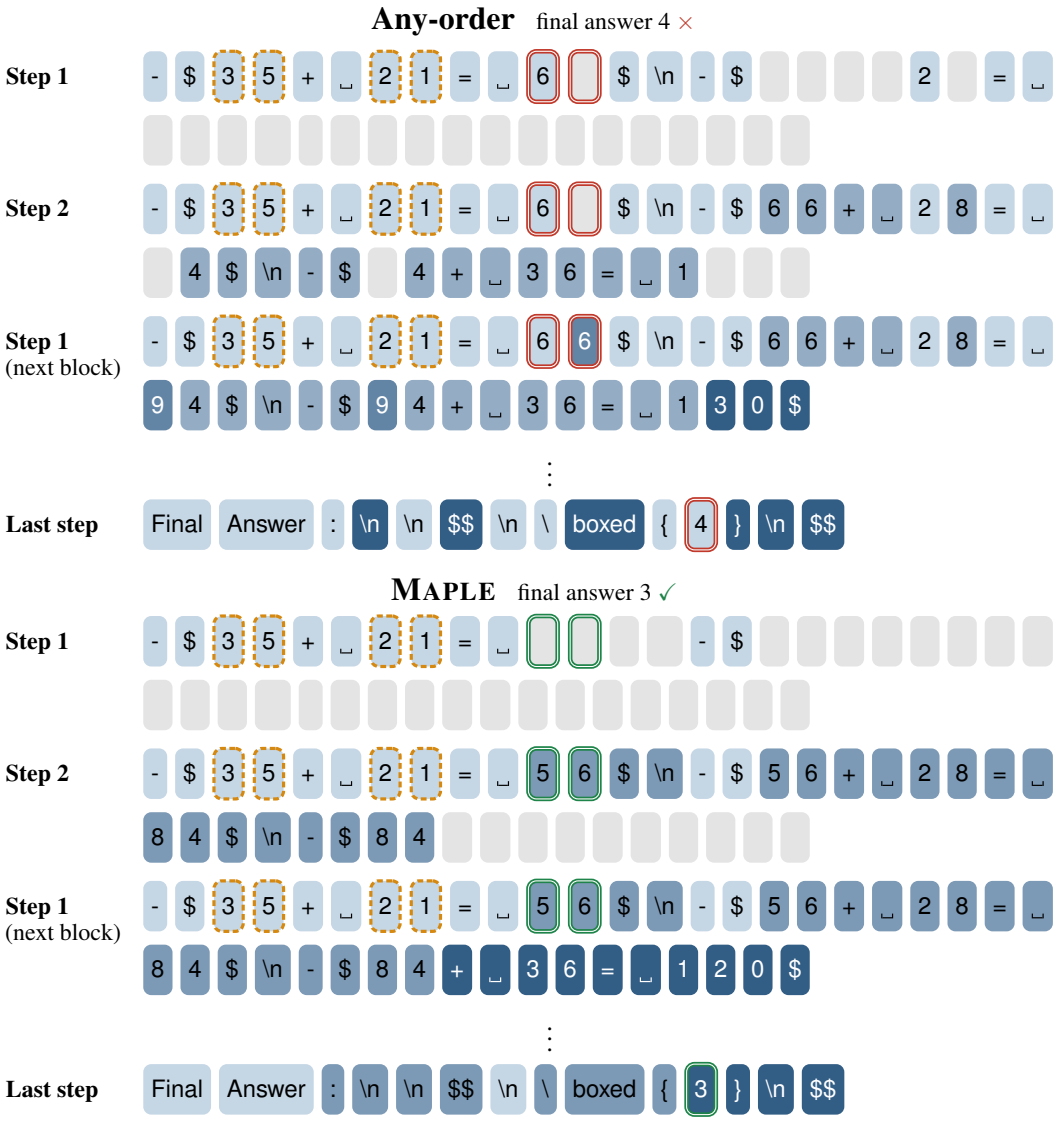


Figure 3: MATH500. Problem: find n with $0 \le n < 9$ and $1 + 12 + 123 + \cdots + 12345678 \equiv n \pmod 9$ (answer 3). Shown: the running sums of the digit sums at the decoding steps given on the left, then the final answer. Steps are counted within the 64-token block being decoded, and rows need not be consecutive steps. Gray : masked; shade: step of commitment, from light (earlier) to dark (later); solid border : tokens that decide the answer; dashed border : tokens they are computed from.

66 instead of 56, which leads to the wrong answer 4. MAPLE commits $35 + 21$ first and its result, 56, one step later, and answers 3.

AMC12-2025 (Figure 4). The any-order model checks each candidate column; for $c = 55$ it has to decide whether $r = 765/9 \approx 85.00$ is an integer. It commits "not integer" before the decimals "00" are revealed, drops this valid square ($765/9 = 85$), and answers 10. MAPLE commits the closed form $r = 14k + 1$ only after the fraction $(126k + 9)/9$ is in place, and answers 11.

OlympiadBench (Figure 5). In the shoelace formula, the any-order model commits the product $(6)(2)$ as 36 in the same step as its two factors, which gives the wrong area 8. MAPLE commits $6 \cdot 2 = 12$ one step after its factors and obtains the area 20.

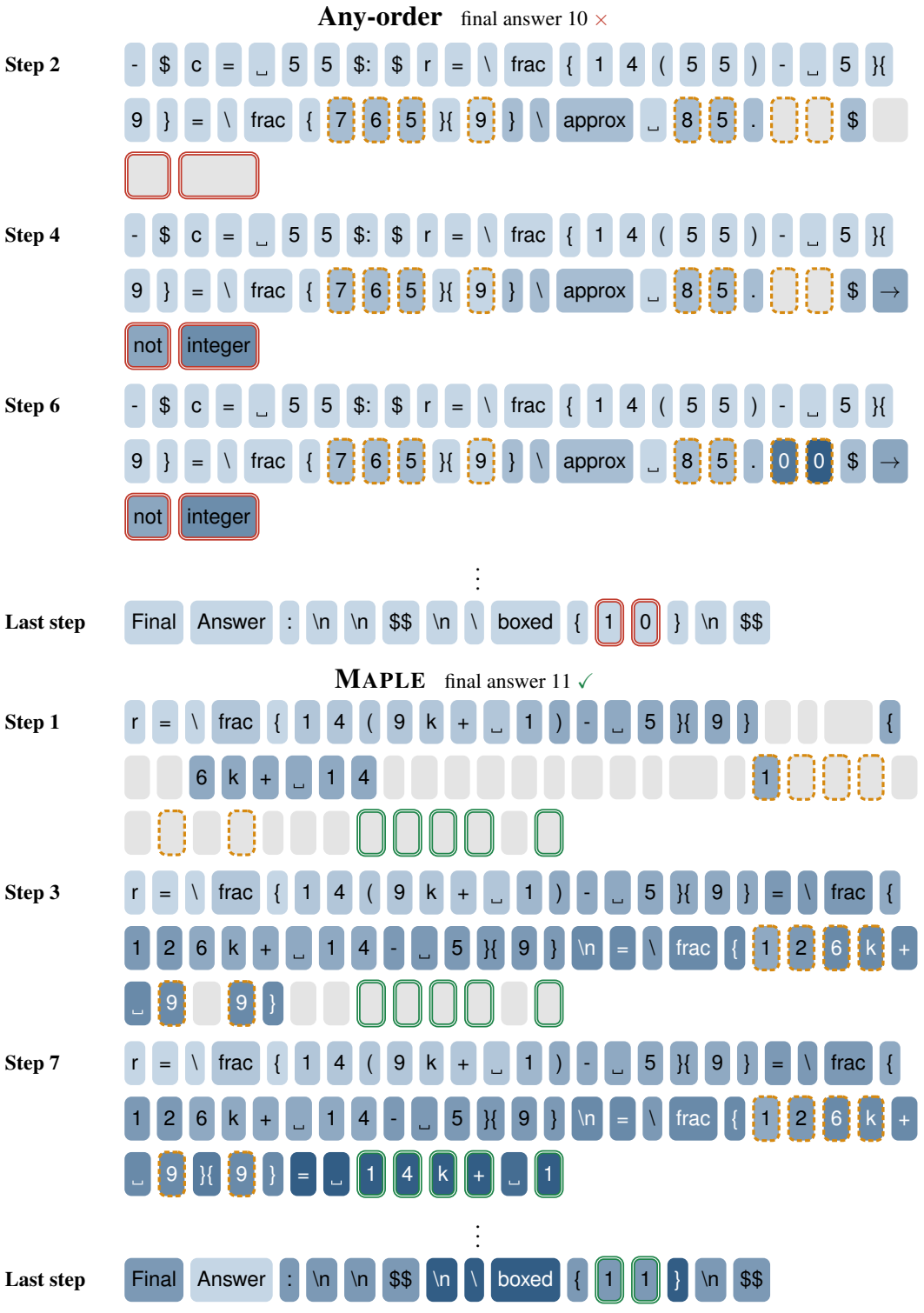


Figure 4: AMC12-2025. Problem: a grid of 141 rows and 91 columns is filled with $1, \ldots, 12{,}831$ once row by row and once column by column; how many squares receive the same number twice (answer 11)? Both models reduce this to the columns $c \equiv 1 \pmod 9$, whose matching row is $r = (14c - 5)/9$. Shown: the check of $c = 55$ (any-order) and the closed form for r (MAPLE), then the final answer. Legend as in Figure 3.

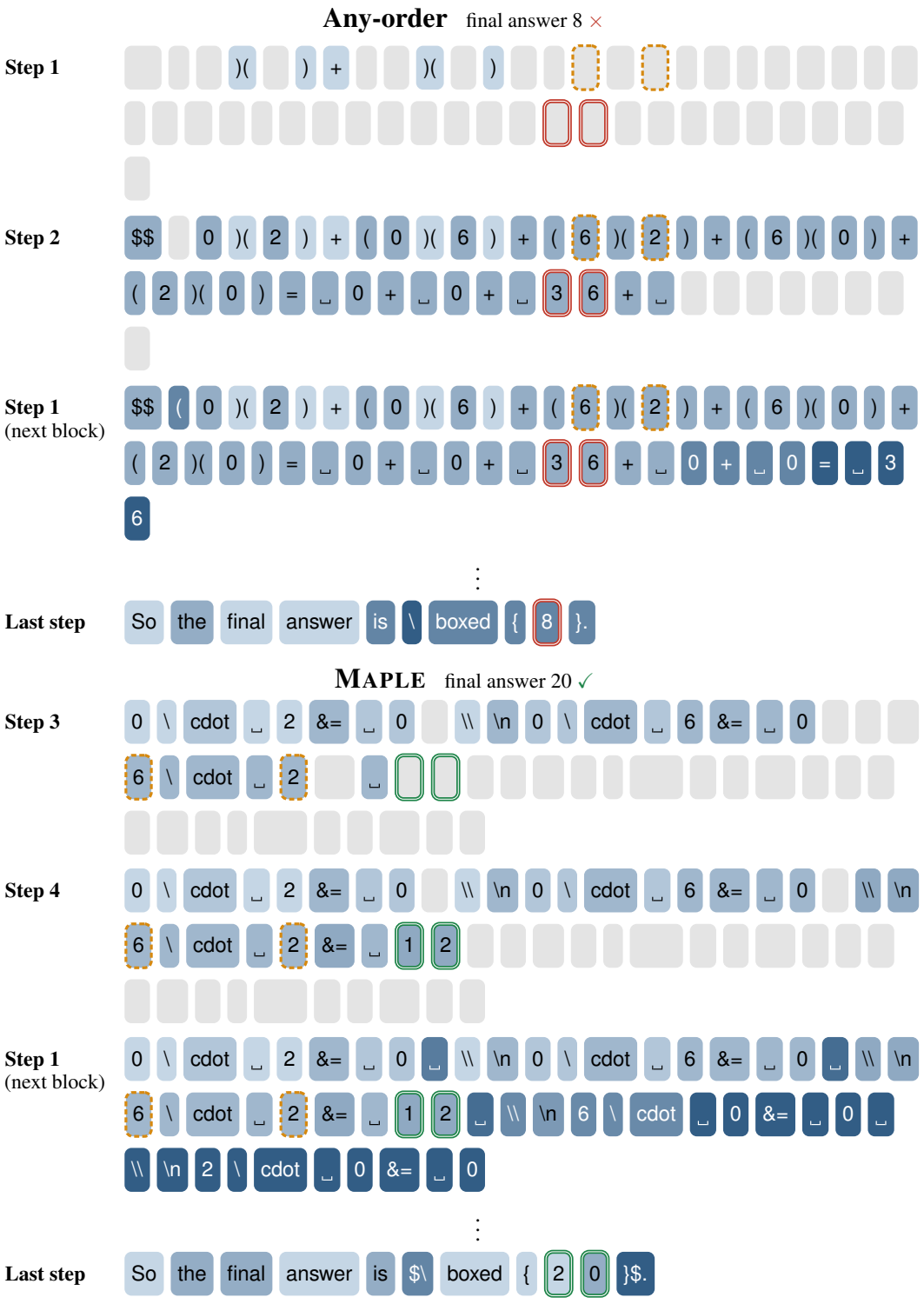


Figure 5: OlympiadBench. Problem: compute the area of pentagon $TNYWR$ with $T = (0, 0)$, $N = (2, 0)$, $Y = (6, 6)$, $W = (2, 6)$, and $R = (0, 2)$ (answer 20). Shown: the products $y_i x_{i+1}$ of the shoelace formula (and, for the any-order model, their sum), then the final answer. Legend as in Figure 3.