A2D2: FINE-TUNING ANY-LENGTH DISCRETE DIFFUSION FOR ADAPTIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion models offer a simple and stable likelihood-based framework for sequence generation, recently extended to any-length settings via token insertion. Principled reward-guided fine-tuning for any-length discrete diffusion, however, remains largely unexplored. We introduce Fine-Tuning **A**ny-Length **D**iscrete **D**iffusion for **A**daptive Decoding (**A2D2**), a unified framework for reward-guided fine-tuning of any-length discrete diffusion models via joint optimization of the insertion and unmasking policies together with a quality-based inference schedule. We derive the Radon–Nikodym derivative for the joint insertion–unmasking path measures, enabling theoretically guaranteed convergence to the intractable reward-tilted sequence distribution without requiring target samples. Building on this, we establish unmasking and insertion quality as tractable approaches for minimizing decoding error and introduce the **Adaptive Joint Decoding (AJD)** loss, which provably yields the optimal path measure that generates the reward-tilted distribution. Empirically, A2D2 improves reward optimization while enhancing generation flexibility and accuracy over prior fixed-length fine-tuning and inference-time guidance methods.

1 INTRODUCTION

Discrete diffusion models (Lou et al., 2024; Austin et al., 2021) have emerged as a leading paradigm for sequence generation, overcoming key limitations of autoregressive models through bidirectional context, any-order generation (Shi et al., 2024), flexible reward guidance (Nisonoff et al., 2025; Tang et al., 2025a), and parallel decoding (Christopher et al., 2025; Ren et al., 2025). These properties have driven strong performance across biological sequence design (Wang et al., 2025c; Gruver et al., 2023), reasoning (Ye et al., 2025; Zhu et al., 2026), and efficient sampling (Holderrieth et al., 2025; Zhu et al., 2025). Among these approaches, **masked discrete diffusion models** (MDMs) (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025; Zheng et al., 2025) stand out for their simple design and stable likelihood-based objective, and have recently been extended to any-length generation by allowing masked-token insertion at arbitrary positions during the diffusion process (Kim et al., 2026a). However, scaling any-length discrete diffusion to reward optimization and fine-tuning remains largely unexplored.

Reward fine-tuning has been studied for fixed-length discrete diffusion (Tang et al., 2025b; Zekri and Boullé, 2025; Wang et al., 2025a; Cao et al., 2025). Any-length models additionally optimize where and how many tokens to insert: length is a design variable for molecules and peptides, while reasoning traces and missing code spans have unknown lengths. Joint insertion and unmasking therefore enlarge both the action space and the set of decoding trajectories. This motivates our **key question**:

Can we efficiently fine-tune any-length discrete diffusion models to sample from an intractable reward-tilted distribution while preserving high generation quality?

To answer this, we introduce Fine-Tuning **A**ny-Length **D**iscrete **D**iffusion for **A**daptive Decoding (**A2D2**), a unified framework for reward-guided fine-tuning of any-length discrete diffusion models via joint optimization of the insertion and unmasking policies and quality-based inference schedule.

Our **main contributions** can be summarized in the following points:

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

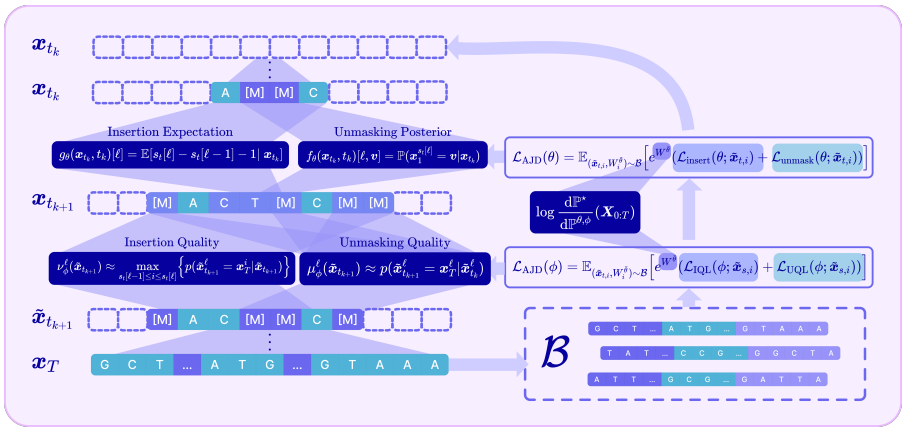

Figure 1: **Fine-Tuning Any-Length Discrete Diffusion for Adaptive Decoding.** A2D2 jointly fine-tunes the insertion and unmasking policies together with a quality classifier for reward-guided adaptive decoding. Sequences are generated via adaptive inference and stored in a replay buffer, then used to compute the Adaptive Joint Decoding loss $\mathcal{L}_{\text{AJD}}$, which weights each sequence's contribution by a policy-based weight $W^{\bar{\theta},\bar{\phi}}$.

1. We specialize the CTMC Radon-Nikodym derivative to joint insertion-unmasking paths, providing the importance weights for a specified proposal measure.
2. We introduce insertion quality, analyze quality under conditional-independence assumptions, and combine policy and quality losses in the **Adaptive Joint Decoding (AJD)** objective.
3. We demonstrate that A2D2 simultaneously optimizes rewards while enhancing generation flexibility and accuracy over prior fixed-length fine-tuning and inference-time guidance approaches.

Related Works A discussion of additional related works is provided in App A.

2 Preliminaries

Masked Discrete Diffusion Models Discrete diffusion generates sequences through a continuous-time Markov chain (CTMC) over $t \in [0,1]$. A *path measure* is its distribution over complete trajectories, specified by an initial distribution and transition rates. The Radon–Nikodym derivative is the likelihood ratio between two such measures, including jump rates and waiting times (Sec. 4.1). **Masked discrete diffusion models** (MDMs) start from fully masked sequences $(\boldsymbol{M})^L$ (Shi et al., 2024; Sahoo et al., 2024; Ou et al., 2025; Zheng et al., 2025); they learn to reconstruct clean data from masked states by minimizing the **denoising cross-entropy (DCE)** loss:

$$\mathcal{L}_{\text{DCE}}(\theta; \boldsymbol{x}_1) := \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)} \mathop{\mathbb{E}}_{\boldsymbol{x}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \left[-\frac{\dot{\alpha}_t}{1-\alpha_t} \sum_{\ell:\boldsymbol{x}_t^\ell=\boldsymbol{M}} \log f_\theta(\boldsymbol{x}_t, t)[\ell, \boldsymbol{x}_1^{s_t[\ell]}] \right] \tag{1}$$

where $\alpha_t : [0,1] \to [0,1]$ is the unmasking schedule bounded by $\alpha_0 = 0$ and $\alpha_1 = 1$ where α_t is the probability that a given token is unmasked at time t in the forward masking process and the unmasking occurs in the reverse process if the time t is greater than the unmasking time $t_u^\ell \sim \dot{\alpha}_t dt$, i.e. $t \geq t_u^\ell$. Under the standard linear schedule, $\alpha_t = t$.

Any-Length Discrete Diffusion To overcome fixed-length initialization, any-length discrete diffusion models (Havasi et al., 2025; Kim et al., 2026a) *insert* tokens along the generation process. We build on the **flexible-length masked diffusion model** framework of Kim et al. (2026a), which defines a *joint stochastic interpolant* over insertion and unmasking.

The forward corruption process gradually masks *and* removes tokens by drawing an insertion time $t_i^\ell \sim \dot{\alpha}_t\, dt$ and unmasking time $t_u^\ell \sim \mathbf{1}[t \geq t_i^\ell] \frac{\dot{\beta} t}{1-\beta t_i^\ell}\, dt$, enforcing that a token is unmasked only after insertion ($t_u^\ell > t_i^\ell$). We track the position of each clean token via $s_t[\ell]$, which is the index

in $\boldsymbol{x}_1$ of the ℓth token in $\boldsymbol{x}_t$, so that $\boldsymbol{x}_t^\ell = \boldsymbol{x}_1^{s_t[\ell]}$. We parameterize both an **unmasking posterior** $f_\theta(\boldsymbol{x}_t, t)[\ell] \in \Delta^{V-1}$ and an **insertion expectation** $g_\theta(\boldsymbol{x}_t, t)[\ell] \in \mathbb{R}_{\geq 0}$, the predicted number of tokens to insert at each gap, and minimize:

$$\mathcal{L}_{\text{insert}}(\theta; \boldsymbol{x}_1) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)\boldsymbol{x}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \left[-\frac{\dot{\alpha}_t}{1-\alpha_t} \sum_{\ell=0}^{\text{len}(\boldsymbol{x}_t)} \phi\left(s_t[\ell] - s_t[\ell-1] - 1, g_\theta(\boldsymbol{x}_t, t)[\ell]\right) \right] \tag{2}$$

$$\mathcal{L}_{\text{unmask}}(\theta; \boldsymbol{x}_1) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)\boldsymbol{x}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \left[-\frac{\dot{\beta}_t}{1-\beta_t} \sum_{\ell:\boldsymbol{x}_t^\ell = \boldsymbol{M}} \log f_\theta(\boldsymbol{x}_t, t)[\ell, \boldsymbol{x}_1^{s_t[\ell]}] \right] \tag{3}$$

where $\phi(x, y) = y - x \log y$ is the scalar Bregman divergence. A detailed background on any-length MDMs is provided in App B.2.

3 Defining the Unmasking and Insertion Quality

In this section, we first introduce *unmasking quality* and relate it to the error accumulated in a single unmasking step (Sec 3.1), then derive an analogous, theoretically grounded notion of *insertion quality* (Sec 3.2).

3.1 Unmasking Quality

Defining Unmasking Quality We define the **unmasking quality** as the probability that the unmasked token is sampled from the unmasking posterior given the context from the rest of the sequence. Concretely, given a clean sequence from the target $\boldsymbol{x}_1 \sim p_{\text{target}}$, a sample along the interpolant $\tilde{\boldsymbol{x}}_t$, a masked position $\tilde{\boldsymbol{x}}_t^\ell = \boldsymbol{M}$, a set of masked tokens $\mathcal{M}_t$ that are unmasked to obtain $\boldsymbol{y}$ where $\boldsymbol{y}^\ell \sim f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell]$, and the unmasking posterior at position ℓ given by $p(\boldsymbol{y}^\ell = \cdot|\tilde{\boldsymbol{x}}_t) = f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell]$, we define the unmasking quality as:

$$\mu_\star^\ell(\boldsymbol{y}) := p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}|\boldsymbol{y}) = f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell, \boldsymbol{x}_1^\ell] \tag{4}$$

which aligns with the definition of the *per-token quality* introduced in (Kim et al., 2026b). When the probability of the true token $p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}|\boldsymbol{y})$ for $\tilde{\boldsymbol{x}}_t^\ell = \boldsymbol{M}$ is high given the unmasked context $\tilde{\boldsymbol{x}}_t^{\text{UM}}$, then the quality is high, and if the probability is low, then the quality is low. To predict the quality of an already unmasked token $\boldsymbol{y}^\ell$ such that it estimates the unmasking posterior defined on its masked state $\boldsymbol{y}^{\mathcal{M}_t \leftarrow \boldsymbol{M}} = \tilde{\boldsymbol{x}}_t$, we train a parameterized model $\mu_\phi : \mathcal{X} \to [0, 1]$ to approximate $\mu_\star^\ell$ given its unmasked state $\boldsymbol{y}$ as:

$$\mu_\phi^\ell(\boldsymbol{y}) \approx p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}|\boldsymbol{y}) \tag{5}$$

During inference, when we do not have access to the ground-truth sequence $\boldsymbol{x}_1$, the unmasking quality determines which tokens are inconsistent and should be re-masked.

Training the Unmasking Quality Predictor To train $\mu_\phi^\ell : \mathcal{X} \to [0, 1]$, we start with a clean sequence from the target distribution $\boldsymbol{x}_1 \sim p_{\text{target}}$ and sample an intermediate state from the joint interpolant $\tilde{\boldsymbol{x}}_t$ by partially masking and removing tokens in $\boldsymbol{x}_1$. Then, we take a single unmasking step to obtain $\boldsymbol{y}$ by replacing a subset $\mathcal{M}$ of tokens in $\tilde{\boldsymbol{x}}_t$ with the clean token sampled from the unmasking posterior $\boldsymbol{y}^\ell \sim f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell]$. Finally, we minimize an **Unmasking Quality Loss** (UQL) defined as:

$$\mathcal{L}_{\text{UQL}}(\phi; \boldsymbol{x}_1) := \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)\tilde{\boldsymbol{x}}_t\sim p_t(\cdot|\boldsymbol{x}_1), \boldsymbol{y}\sim p_{s|t}(\cdot|\tilde{\boldsymbol{x}}_t)} \left[\sum_{\ell\in\mathcal{M}} \text{BCE}\left(\mathbf{1}[\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}], \mu_\phi^\ell(\boldsymbol{y})\right) \right] \tag{6}$$

where BCE denotes the **binary cross-entropy** loss defined as $\text{BCE}(b, c) = -b \log c - (1-b) \log(1-c)$ for $b \in \{0, 1\}$ and $c \in [0, 1]$. The *unique minimizer* of $\mathcal{L}_{\text{UQL}}(\phi)$ is exactly the unmasking quality.

Proposition 3.1 (Unique Minimizer of Unmasking Quality Loss)**.** *The unique minimizer of the unmasking quality loss $\mathcal{L}_{UQL}(\phi)$ is the true unmasking quality:*

$$\mu_\star = \mu_{\phi^\star}, \quad \textit{where} \quad \phi^\star = \arg\min_\phi \mathcal{L}_{UQL}(\phi) \tag{7}$$

The proof is stated in App C.2. We note that this coincides with the loss defined in Kim et al. (2026b) for parameterizing the unmasking quality of fixed-length MDMs.

Compounding Parallelization Error of Unmasking Steps For **unmasking**, the compounding parallelization error is the KL divergence between the true joint distribution over tokens unmasked in a step and the product of their marginals (Park et al., 2025), defined as the gap between sequential single-token unmasking and the factorized approximation made when unmasking in parallel.

Definition 3.1 (Unmasking Compounding Parallelization Error)**.** *Consider a unmasking step from state $\boldsymbol{x}_s$ to $\boldsymbol{x}_t$, where $\{\boldsymbol{x}_t^{\ell_k}\}_{k=1}^K$ denotes the set of K tokens unmasked in parallel at time t. The unmasking CPE over the transition $s \to t$ is defined as:*

$$\mathcal{E}_{CPE}^{unmsk}(s \to t) = \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s}\left[D_{KL}\left(p(\boldsymbol{x}_t^{\ell_1}, \ldots, \boldsymbol{x}_t^{\ell_K} | \boldsymbol{x}_s) \middle\| \prod_{k=1}^K p(\boldsymbol{x}_t^{\ell_k} | \boldsymbol{x}_s) \right) \right] \tag{8}$$

where $\mathbb{P}_s$ is the marginal distribution of sequences at time s.

Optimal Parallel Unmasking as Maximizing the Unmasking Quality We now show that unmasking quality provides an upper bound for the probability of successful parallel unmasking.

Proposition 3.2 (Relationship Between Unmasking Quality and Parallel Unmasking)**.** *Assume that in a parallel unmasking step on indices $\mathcal{M}_t = \{\ell_k\}_{k=1}^K$, the unmasked tokens are conditionally independent given the unchanged context $\bar{\boldsymbol{x}}_s$ and the current state $\boldsymbol{x}_s$, i.e. $p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^K p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s)$. If we define the unmasking quality $\mu_\star^{\ell_k}(\boldsymbol{x}_t) := p(\boldsymbol{x}_t^{\ell_k} = \boldsymbol{x}_1^{\ell_k} | \boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}})$ and note that $p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = p(\boldsymbol{x}_t^{\ell_k} | \boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}})$, then*

$$p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^K \mu_\star^{\ell_k}(\boldsymbol{x}_t) \tag{9}$$

The proof is deferred to App C.4. This establishes the theoretical grounding for our adaptive unmasking strategy, which remasks low-quality tokens in $\boldsymbol{x}_t$ to ensure only mutually high-quality tokens are unmasked in parallel, which effectively maximizes the probability of optimal parallel unmasking.

3.2 Insertion Quality

Defining Insertion Quality It is natural to define the **insertion quality** as the probability that the insertion is likely to be decoded into a true token in the corresponding gap of the target sequence. Concretely, given a clean sequence from the target distribution $\boldsymbol{x}_1 \sim p_{\text{target}}$ and a sample along the interpolant $\tilde{\boldsymbol{x}}_t$, we define quality of a mask inserted between positions $\ell - 1$ and ℓ, denoted $\boldsymbol{y} := \tilde{\boldsymbol{x}}_t^{\triangleleft_\ell \boldsymbol{M}}$, as:

$$\nu_\star(\boldsymbol{y}) := \max_{s_t[\ell-1] < i < s_t[\ell]} \left\{ p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^i | \boldsymbol{y}) \right\} \tag{10}$$

which returns the highest probability of any ground truth token between positions $s_t[\ell - 1]$ and $s_t[\ell]$ in the clean sequence $\boldsymbol{x}_1$ being predicted by the unmasking posterior $p(\boldsymbol{y}^i = \cdot | \boldsymbol{y}) = f_\theta(\boldsymbol{y}, t)[i]$ at the newly inserted mask token $\boldsymbol{y}^i = (\tilde{\boldsymbol{x}}_t^{\triangleleft_\ell \boldsymbol{M}})^i = \boldsymbol{M}$. Alternatively, we can define the set of tokens $\mathcal{S}_\ell := \{\boldsymbol{x}_1^i : s_t[\ell - 1] < i < s_t[\ell]\}$ in the gap between positions $\ell - 1$ and ℓ and the insertion quality

as the sum of the probabilities of the tokens in $\mathcal{S}_\ell$, given by:

$$\nu^\ell_\star(\boldsymbol{y}) := \sum_{\boldsymbol{v}\in\mathcal{S}_\ell} p(\boldsymbol{y}^\ell = \boldsymbol{v}|\boldsymbol{y}) = p(\boldsymbol{y}^\ell \in \mathcal{S}_\ell|\boldsymbol{y}) \tag{11}$$

which is high when several potential tokens have a **high potential** to be unmasked to one of the tokens within the gap. Note that when there is **no token** between positions $s_t[\ell-1]$ and $s_t[\ell]$ in $\boldsymbol{x}_1$, the quality is zero. Since we have no access to the ground truth sequence at inference, we aim to train a parameterized model ν_ϕ that takes only the sequence *after the insertion step* $\boldsymbol{y}$ as input and approximates the quality of the insertion:

$$\nu^i_\phi(\boldsymbol{y}) \approx \sum_{\boldsymbol{v}\in\mathcal{S}_\ell} p(\boldsymbol{y}^\ell = \boldsymbol{v}|\boldsymbol{y}) \in [0,1] \tag{12}$$

Unlike fixed-length token quality, insertion quality scores whether a candidate slot can decode to a token in a possibly empty target gap. Its soft target trains the insertion predictor, and low-scoring inserted masks are removed during adaptive inference.

Training the Insertion Quality Predictor To train the insertion quality predictor ν_ϕ, we sample a clean sequence $\boldsymbol{x}_1 \sim p_{\text{target}}$ and apply the joint interpolant to obtain $\tilde{\boldsymbol{x}}_t$ by partially masking and removing tokens. We then sample insertion counts at each gap using $g_\theta(\tilde{\boldsymbol{x}}_t, t)$ and insert the corresponding masks to obtain $\boldsymbol{y}$. Let $\mathcal{I} = \{i \mid \boldsymbol{y}^i \text{ is a newly inserted mask}\}$ denote their positions. For each $i \in \mathcal{I}$, we compute the unmasking posterior $f_\theta(\boldsymbol{y}, t)[i]$ and use it to construct the insertion-quality target. Finally, we minimize the **Insertion Quality Loss** (IQL), defined as:

$$\mathcal{L}_{\text{IQL}}(\phi; \boldsymbol{x}_1) := \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)} \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \mathop{\mathbb{E}}_{\boldsymbol{y}\sim p_{s|t}(\cdot|\tilde{\boldsymbol{x}}_t)} \left[\sum_{i\in\mathcal{I}} \text{BCE}\big(\nu^i_\star(\boldsymbol{y}), \nu^i_\phi(\boldsymbol{y})\big)\right] \tag{13}$$

where BCE denotes the **Bernoulli cross-entropy** loss for two values in defined as $\text{BCE}(b,c) = -b\log c - (1-b)\log(1-c)$ for two values $b, c \in [0,1]$. If there are multiple masks $(\boldsymbol{y}^{i_1}, \ldots, \boldsymbol{y}^{i_D})$ inserted in a gap $(\boldsymbol{x}_1^{s_t[\ell-1]+1}, \ldots, \boldsymbol{x}_1^{s_t[\ell]-1})$, then for each i_d, we sum over the probabilities over the range $s_t[\ell-1]+d$ to $s_t[\ell]-1-(D-d)$. We show that the minimizer of the insertion quality loss $\mathcal{L}_{\text{IQL}}(\phi)$ is the true insertion quality $\nu_\star$ defined in (11) with proof in App C.3.

Proposition 3.3 (Minimizer of Insertion Quality Loss)**.** *The unique minimizer of the insertion quality loss $\mathcal{L}_{IQL}(\phi)$ is the true insertion quality:*

$$\nu_\star = \nu_{\phi^\star}, \quad \textit{where} \quad \phi^\star = \arg\min_\phi \mathcal{L}_{IQL}(\phi) \tag{14}$$

Optimizing Reconstruction Likelihood by Maximizing Insertion Quality To theoretically justify maximizing insertion quality, we prove that it provides an upper bound for the probability of reconstructing a clean sequence $\boldsymbol{x}_1 \sim p_{\text{target}}$.

Proposition 3.4 (Insertion Quality as an Upper Bound on Reconstruction via Insertions)**.** *Consider a clean target sequence $\boldsymbol{x}_1 \sim p_{target}$ and an intermediate sequence $\boldsymbol{x}_s \sim p_s(\cdot|\boldsymbol{x}_1)$. Given a set of indices of inserted masks at each gap $\mathcal{I}_t$ which yields $\boldsymbol{x}_t^{\mathcal{I}_t}$, the probability of reconstructing $\boldsymbol{x}_1$ is upper-bounded by the product of insertion qualities:*

$$p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t}|\boldsymbol{x}_t) \le \prod_{i\in\mathcal{I}_t} \nu^i_\star(\boldsymbol{x}_t) \approx \prod_{i\in\mathcal{I}_t} \nu^i_\phi(\boldsymbol{x}_t) \tag{15}$$

where the unmasking posterior for each inserted mask is conditionally independent given $\boldsymbol{x}_t$.

We defer the proof to App C.5. This justifies maximizing insertion quality as a surrogate for improving reconstruction quality.

4 A2D2: FINE-TUNING ANY-LENGTH DISCRETE DIFFUSION FOR ADAPTIVE DECODING

In this section, we introduce Fine-Tuning **A**ny-Length **D**iscrete **D**iffusion for **A**daptive Decoding (**A2D2**), a novel framework tailored for any-length masked diffusion models that jointly optimizes the parameterized quality predictors and policy to sample from the reward-tilted data distribution.

4.1 ADAPTIVE JOINT DECODING LOSS

Optimal Reward-Tilted Path Measure Since the path measure $\mathbb{P}^v$ of any-length MDMs consists of both insertion steps *and* unmasking steps, which form the joint CTMC with rates $\boldsymbol{R}_t^v$ and $\boldsymbol{Q}_t^v$, respectively, we aim to define the tilted path measure $\mathbb{P}^\star$ that optimally samples from the reward-tilted distribution p_{target}. Concretely, we define the tilted path measure as:

$$\mathbb{P}^\star(\boldsymbol{X}_{0:1}) := \frac{1}{Z}\mathbb{P}^{\text{pre}}(\boldsymbol{X}_{0:1}) \exp\left(\frac{r(\boldsymbol{X}_1)}{\alpha}\right), \quad \mathbb{P}_1^\star(\boldsymbol{x}) = \frac{1}{Z} p_{\text{data}}(\boldsymbol{x}) \exp\left(\frac{r(\boldsymbol{x})}{\alpha}\right) =: p_{\text{target}}(\boldsymbol{x}) \tag{16}$$

where $\boldsymbol{X}_{0:1} = (\boldsymbol{X}_t)_{t\in[0,1]}$ is a joint CTMC induced by a path measure and $\mathbb{P}^{\text{pre}}$ is the reference path measure induced by the pre-trained insertion rate $\boldsymbol{R}_t^{\text{pre}}$ and unmasking rate $\boldsymbol{Q}_t^{\text{pre}}$. Given parameterized insertion and unmasking rates $\boldsymbol{R}_t^\theta$ and $\boldsymbol{Q}_t^\theta$, we can optimize θ such that the path measure $\mathbb{P}^\theta$ matches $\mathbb{P}^\star$ by maximizing the following **entropy-regularized reward optimization** problem (Uehara et al., 2024):

$$\max_\theta \mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}^\theta}\left[r(\boldsymbol{X}_1)\right] - \alpha D_{\text{KL}}(\mathbb{P}^\theta \| \mathbb{P}^{\text{pre}}) \tag{17}$$

which is uniquely maximized when $\mathbb{P}^\theta = \mathbb{P}^\star$. For any-length joint CTMCs, the KL-divergence takes a unique form, as derived in App C.7.

Weighted Cross-Entropy Objective We first consider the KL objective $D_{\text{KL}}(\mathbb{P}^{\theta,\phi}\|\mathbb{P}^\star)$. While its minimizer indeed matches the optimal measure $\mathbb{P}^\star$, it is difficult to optimize due to the expectation over $\mathbb{P}^{\theta,\phi}$, which changes as the model is updated. Instead, we use the **cross-entropy loss** which takes the expectation over the **fixed** target path measure $\mathbb{P}^\star$:

$$\mathcal{F}_{\text{CE}}(\mathbb{P}^{\theta,\phi}, \mathbb{P}^\star) := D_{\text{KL}}(\mathbb{P}^\star\|\mathbb{P}^{\theta,\phi}) = \mathbb{E}_{\mathbb{P}^\star}\left[\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] = \mathbb{E}_{\mathbb{P}^v}\left[\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v} \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] \tag{18}$$

where the final equality allows us to sample trajectories from any arbitrary path measure $\mathbb{P}^v$ and **reweight** them such that they approximate the target path measure $\mathbb{P}^\star$ at the infinite sampling limit. The reweighting term $\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}$ is the Radon-Nikodym derivative (RND) between the two path measures. We derive the full tractable form of the RND in Prop 4.1.

Proposition 4.1 (Radon-Nikodym Derivative of Parameterized Rates)**.** *Let the fine-tuned unmasking rate be $f^v(\boldsymbol{x}_t, t)[\ell] \in \Delta^{V-1}$ and the insertion rate be $g^v(\boldsymbol{x}_t, t)[\ell] \in \mathbb{R}_{\geq 0}$ that generates the path measure $\mathbb{P}^v$. Then, given optimal rates $f^{pre}(\boldsymbol{x}_t, t)$ and $g^{pre}(\boldsymbol{x}_t, t)$ and reward function $r : \mathcal{X} \to \mathbb{R}$, the log RND between the optimal joint CTMC and the fine-tuned CTMC over the trajectory $\boldsymbol{X}_{0:1} = (\boldsymbol{X}_t)_{t\in[0,1]}$ is defined as:*

$$\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) = \frac{r(\boldsymbol{X}_1)}{\alpha} - \log Z + \sum_{t_u:\boldsymbol{X}_{t_u}\neq\boldsymbol{X}_{t_u-}} \sum_{\ell:\boldsymbol{X}^\ell_{t_u}\neq\boldsymbol{X}^\ell_{t_u-}} \log \frac{f^{pre}(\boldsymbol{X}_{t_u}, t_u)[\ell, \boldsymbol{v}]}{f^v(\boldsymbol{X}_{t_u}, t_u)[\ell, \boldsymbol{v}]}$$

$$+ \sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}} \sum_{\ell:\boldsymbol{X}^\ell_{t_i}\neq\boldsymbol{X}^\ell_{t_i-}} \log \frac{g^{pre}(\boldsymbol{X}_{t_i}, t_i)[\ell]}{g^v(\boldsymbol{X}_{t_i}, t_i)[\ell]} + \int_0^1 \frac{\dot{\alpha}_t}{1-\alpha_t}\left(\sum_\ell (g^v - g^{pre})(\boldsymbol{X}_t, t)[\ell]\right) dt \tag{19}$$

where $t_i \in [0,1]$ denotes the times of insertion events and $t_u \in [0,1]$ denotes the times of unmasking events, with t_i^- and t_u^- being the left limits denoting the pre-jump state in the CTMC.

The proof is given in App C.8. Then, defining $W^v(\boldsymbol{X}_1) := \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}(\boldsymbol{X}_{0:1})$ as an unnormalized log importance weight, we can write the cross-entropy loss as $\mathcal{F}_{\text{CE}}(\mathbb{P}^{\theta,\phi}, \mathbb{P}^\star) = \mathbb{E}_{\mathbb{P}^v}[\frac{1}{Z} e^{W^v} \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}]$.

Adaptive Joint Decoding Loss Now, we derive our **Adaptive Joint Decoding (AJD)** loss, which allows us to optimize the cross-entropy loss without computing the inner RND $\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}$ over trajectories after each update to the model parameters.

$$\begin{aligned}
&\min_{\theta,\phi} \mathbb{E}_{\mathbb{P}^v}\left[\frac{1}{Z}e^{W^v}\log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] = \min_{\theta,\phi} \mathbb{E}_{\mathbb{P}^v}\left[\frac{1}{Z}e^{W^v}[-\log\mathbb{P}^{\theta,\phi}]\right] \\
&= \min_{\theta,\phi} \mathbb{E}_{\mathbb{P}^v}\left[\frac{1}{Z}e^{W^v}\left[\sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}} -\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_i}|\boldsymbol{X}_{t_i^-}) + \sum_{t_u:\boldsymbol{X}_{t_u}\neq\boldsymbol{X}_{t_u-}} -\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_u}|\boldsymbol{X}_{t_u^-})\right]\right]
\end{aligned} \tag{20}$$

where the first equality follows from dropping the $\log\mathbb{P}^\star$ term independent of θ,ϕ, and the second equality follows from decomposing the probability path into the sum of probabilities of each insertion and unmasking step at times t_i and t_u, respectively. Rather than explicitly computing the sum over the full trajectory, we observe that each inner term can be reframed as an expectation over samples from the interpolant $\tilde{\boldsymbol{x}}_t \sim p_t(\cdot|\boldsymbol{X}_1)$ given a clean sample $\boldsymbol{X}_1$ from $\boldsymbol{X}_{0:1}\sim\mathbb{P}^v$. Given the clean sample $\boldsymbol{x}_1$, the optimal f_θ, g_θ, μ_ϕ, and ν_ϕ can be obtained by minimizing $\mathcal{L}_{\text{unmask}}(\theta;\boldsymbol{x}_1)$ (3), $\mathcal{L}_{\text{insert}}(\theta;\boldsymbol{x}_1)$ (2), $\mathcal{L}_{\text{UQL}}(\phi;\boldsymbol{x}_1)$ (6), and $\mathcal{L}_{\text{IQL}}(\phi;\boldsymbol{x}_1)$ (13), respectively. This yields our **Adaptive Joint Decoding (AJD)** loss, defined as:

$$\mathcal{L}_{\text{AJD}}(\theta,\phi) := \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}^v}\left[\frac{1}{Z}\mathrm{e}^{W^v}\left[\mathcal{L}_{\text{unmask}}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{\text{insert}}(\theta;\boldsymbol{x}_1)+\lambda_{\text{quality}}(\mathcal{L}_{\text{UQL}}(\phi;\boldsymbol{X}_1)+\mathcal{L}_{\text{IQL}}(\phi;\boldsymbol{X}_1))\right]\right] \tag{21}$$

where the normalizing constant Z is approximated as $Z\approx\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}^v}\left[\mathrm{e}^{W^v}\right]$. This loss can be seen as an extension of the weighted denoising cross-entropy loss used for fixed-length discrete diffusion sampling (Zhu et al., 2025) and fine-tuning (Tang et al., 2025b). This unique minimizer of the AJD loss is the optimal unmasking generator $\boldsymbol{Q}^\star$ and insertion generator $\boldsymbol{R}^\star$ of the reward-tilted path measure $\mathbb{P}^\star$, with proof deferred to App C.8.

4.2 Fine-Tuning Any-Order Discrete Diffusion

Off-Policy Reinforcement Learning To optimize the AJD loss, we leverage an **off-policy RL** strategy with the fixed path measure $\mathbb{P}^v := \mathbb{P}^{\bar{\theta},\bar{\phi}}$ defined by detached parameters $\bar{\theta} := \texttt{stopgrad}(\theta)$ and $\bar{\phi} := \texttt{stopgrad}(\phi)$. Each fine-tuning iteration: **(1)** we sample a batch of B sequences $\boldsymbol{x}_1$ while computing their log RND $W^{\bar{\theta},\bar{\phi}}$ and store them in a replay buffer $\mathcal{B}=\{(\boldsymbol{x}_{1,i},W^{\bar{\theta},\bar{\phi}})\}_{i=1}^B$, **(2)** for each $\boldsymbol{x}_{1,i}$, we sample R intermediate sequences from the interpolant $\tilde{\boldsymbol{x}}_{t,j}\sim p_t(\cdot|\boldsymbol{x}_{1,i})$, and **(3)** we compute the AJD loss (21) and update the parameters θ,ϕ.

Alternating Optimization of Policy and Quality Predictors Since the number of parameters θ in the insertion and unmasking policy is orders of magnitude larger than the number of parameters ϕ in the quality prediction heads, optimizing both models simultaneously results in unstable training and sub-optimal convergence of the quality prediction model. Following prior work, we alternate between updating θ for N_{alt} iterations with ϕ frozen and updating ϕ for N_{alt} iterations with θ frozen (Zhou et al., 2025). Since the quality heads are initialized from scratch, we sample the replay buffer using only the policy model θ for N_{warmup} iterations. Beyond stabilizing training, this alternating schedule acts as an implicit form of loss balancing: because θ and ϕ are never updated in the same pass, the relative weighting λ_{quality} between the policy losses ($\mathcal{L}_{\text{unmask}},\mathcal{L}_{\text{insert}}$) and the quality losses ($\mathcal{L}_{\text{UQL}},\mathcal{L}_{\text{IQL}}$) no longer needs to be tuned. Full pseudo-code for fine-tuning with A2D2 is given in Algorithm 1.

Adaptive Inference with A2D2 Adaptive inference with the unmasking and insertion policy f_θ,g_θ and quality predictors μ_ϕ,ν_ϕ starts from an empty sequence of pad tokens $\boldsymbol{x}_0$ and at each discrete time step $[t_k,t_{k+1}]$, performs:

(i) **Adaptive Unmasking**: We sample a subset of mask tokens $\mathcal{M}$ to unmask via the parameterized unmasking posterior $f_\theta(\boldsymbol{x}_t,t)[\ell]$. Then, we predict the unmasking quality for each token $\mu_\phi(\boldsymbol{x}_t,t)[\ell]$ and re-mask low-quality tokens that either fall below a threshold or until the expected number of masked tokens at time t is reached.

Table 1: **Drug-like small molecule design results.** Metrics were computed for 1000 generated sequences, and the mean and standard deviations were computed across 3 seeds.

Method	Validity (% total)	Uniqueness (% valid)	Quality (% total)	QED (↑)	Synthetic Accessibility (↓)	Diversity (↑)
GenMol w/o confidence	$98.467_{\pm 0.094}$	$99.763_{\pm 0.048}$	$73.667_{\pm 1.159}$	$0.763_{\pm 0.002}$	$3.024_{\pm 0.017}$	$0.859_{\pm 0.000}$
GenMol w/ confidence	$99.733_{\pm 0.047}$	$99.532_{\pm 0.125}$	$84.000_{\pm 0.566}$	$0.813_{\pm 0.004}$	$2.881_{\pm 0.020}$	$0.817_{\pm 0.000}$
Pre-trained (Any Length)	$95.800_{\pm 0.294}$	$99.582_{\pm 0.086}$	$44.167_{\pm 0.574}$	$0.641_{\pm 0.002}$	$3.401_{\pm 0.032}$	$0.908_{\pm 0.000}$
A2D2 w/o quality	$96.400_{\pm 0.535}$	$62.484_{\pm 0.452}$	$76.067_{\pm 0.896}$	$0.795_{\pm 0.005}$	$2.306_{\pm 0.024}$	$0.730_{\pm 0.003}$
A2D2 w/o insertion quality	$92.200_{\pm 0.082}$	$93.746_{\pm 0.335}$	$71.067_{\pm 0.411}$	$0.752_{\pm 0.003}$	$2.654_{\pm 0.014}$	$0.832_{\pm 0.001}$
A2D2 w/o unmasking quality	$98.600_{\pm 0.082}$	$85.801_{\pm 0.387}$	$82.367_{\pm 0.736}$	$0.818_{\pm 0.003}$	$2.753_{\pm 0.026}$	$0.789_{\pm 0.002}$
A2D2 w/ both quality	$94.533_{\pm 0.665}$	$93.695_{\pm 0.925}$	$71.300_{\pm 0.852}$	$0.762_{\pm 0.004}$	$2.870_{\pm 0.018}$	$0.843_{\pm 0.001}$

Table 2: **Multi-objective peptide design.** Mean $\pm$ standard deviation over 3 seeds, with 1000 samples per run except guidance (100 Pareto-optimal samples after 100 iterations). Off-policy RL uses 500 fine-tuning iterations. Fixed/Any denote fixed-/any-length MDMs; w/o quality omits both quality predictors. *Guidance retains only valid sequences, so validity is not comparable.

Target Protein	Method	Validity (%)	Binding Affinity (↑)	Solubility (↑)	Non-hemolysis (↑)	Non-fouling (↑)	Permeability (↑)
TfR	Pre-trained (Fixed)	$32.446_{\pm 0.662}$	$8.776_{\pm 0.023}$	$0.707_{\pm 0.001}$	$0.901_{\pm 0.003}$	$0.215_{\pm 0.001}$	$-7.145_{\pm 0.003}$
	Pre-trained (Any)	$10.064_{\pm 0.360}$	$7.756_{\pm 0.015}$	$0.689_{\pm 0.010}$	$0.851_{\pm 0.001}$	$0.169_{\pm 0.004}$	$-7.185_{\pm 0.009}$
	Multi-Objective Guidance (Fixed)	N/A*	$9.043_{\pm 0.168}$	$0.717_{\pm 0.004}$	$0.897_{\pm 0.009}$	$0.206_{\pm 0.001}$	$-7.126_{\pm 0.004}$
	Off-Policy RL (Fixed)	$33.433_{\pm 1.190}$	$10.614_{\pm 0.011}$	$0.655_{\pm 0.004}$	$0.810_{\pm 0.002}$	$0.224_{\pm 0.002}$	$-7.109_{\pm 0.005}$
	A2D2 w/o quality	$41.267_{\pm 1.520}$	$10.241_{\pm 0.012}$	$0.792_{\pm 0.011}$	$0.831_{\pm 0.004}$	$0.096_{\pm 0.001}$	$-6.102_{\pm 0.006}$
	A2D2 w/ both quality	$48.600_{\pm 1.445}$	$10.190_{\pm 0.033}$	$0.776_{\pm 0.003}$	$0.831_{\pm 0.003}$	$0.118_{\pm 0.002}$	$-6.338_{\pm 0.003}$
GLP-1R	Pre-trained (Fixed)	$32.446_{\pm 0.662}$	$8.781_{\pm 0.018}$	$0.707_{\pm 0.001}$	$0.901_{\pm 0.003}$	$0.215_{\pm 0.001}$	$-7.145_{\pm 0.003}$
	Pre-trained (Any)	$10.064_{\pm 0.360}$	$8.008_{\pm 0.019}$	$0.689_{\pm 0.010}$	$0.851_{\pm 0.001}$	$0.169_{\pm 0.004}$	$-7.185_{\pm 0.009}$
	Multi-Objective Guidance (Fixed)	N/A*	$8.897_{\pm 0.041}$	$0.721_{\pm 0.010}$	$0.896_{\pm 0.015}$	$0.214_{\pm 0.014}$	$-7.111_{\pm 0.025}$
	Off-Policy RL (Fixed)	$48.767_{\pm 0.492}$	$9.377_{\pm 0.025}$	$0.684_{\pm 0.003}$	$0.804_{\pm 0.010}$	$0.247_{\pm 0.002}$	$-7.019_{\pm 0.009}$
	A2D2 w/o quality	$25.033_{\pm 2.014}$	$9.698_{\pm 0.003}$	$0.830_{\pm 0.003}$	$0.849_{\pm 0.004}$	$0.618_{\pm 0.006}$	$-7.272_{\pm 0.003}$
	A2D2 w/ both quality	$29.167_{\pm 1.634}$	$9.520_{\pm 0.017}$	$0.854_{\pm 0.002}$	$0.840_{\pm 0.002}$	$0.601_{\pm 0.007}$	$-7.209_{\pm 0.008}$

(ii) **Adaptive Insertion**: We insert a set of masks $\mathcal{I}$ according by sampling insertion counts $I_t^\ell \sim \text{Poisson}(g_\theta(\boldsymbol{x}_t, t)[\ell] \cdot \Delta t)$ for each gap. Then, we predict the insertion quality $\nu_\phi(\boldsymbol{x}_t, t)[i]$ for each inserted mask and subsequently remove low-quality masks.

We stop when no masks remain, and the insertion expectation falls below 0.5 or when the total number of time steps is reached. The full inference procedure is detailed in Algorithm 2.

5 Experiments

We evaluate **A2D2** on three reward-guided sequence generation tasks where any-order, any-length discrete diffusion is particularly well-suited: drug-like small-molecule design (Sec 5.1), multi-objective therapeutic peptide generation (Sec 5.2), and language reasoning for math word problems and code infilling (Sec 5.3).

5.1 Drug-Like Small Molecule Design

Setup and Baselines We pre-train an any-length MDM on the SAFE dataset (Noutahi et al., 2024), containing $\sim$ 950M molecules from ZINC (Irwin et al., 2012) and Unichem (Chambers et al., 2013) in SAFE notation ($V = 1880$). We fine-tune with **A2D2** using QED and SA as rewards, and evaluate *quality* (fraction of valid, unique, drug-like, and synthesizable molecules), *validity*, *uniqueness*, and *diversity* (mean Tanimoto distance between Morgan fingerprints). Following Jin et al. (2020), molecules are drug-like if QED (Bickerton et al., 2012) > 0.6 and synthesizable if SA (Ertl and Schuffenhauer, 2009) ≤ 4 (Noutahi et al., 2024; Lee et al., 2025). We compare against the pre-trained MDM and **GenMol** (Lee et al., 2025), the state-of-the-art *fixed-length* masked diffusion model for SAFE generation. Confidence-based sampling with GenMol uses a confidence score computed as the sum of the log-probability the model assigned to its own prediction with scaled Gumbel noise. Additional experiment details and ablations in App D.

Results Fine-tuning any-length MDMs with our AJD-weighted loss yields substantial improvements in molecular rewards over the pre-trained any-length baseline, raising QED from 0.641 to 0.762 and lowering SA from 3.40 to 2.87, while improving quality from 44.2% to 71.3% (Table 1). These gains close the gap to GenMol (Lee et al., 2025) in validity, diversity, and quality, while operating in the strictly more general any-length setting.

5.2 Multi-Objective Therapeutic Peptide Generation

Setup and Baselines We pre-train an any-length MDM on 11M peptide SMILES with the dataset used in Tang et al. (2025a). We fine-tune with A2D2 on five therapeutic properties: binding affinity to a target protein, solubility, non-hemolysis, non-fouling, and permeability, and evaluate validity using the `SMILES2PEPTIDE` decoder (Tang et al., 2025a). As no prior work addresses inference-time guidance or fine-tuning for any-length discrete diffusion, we compare against fixed-length masked diffusion baselines: **Multi-Objective Guidance (Fixed)** (Tang et al., 2025a) and **Off-Policy RL (Fixed)** (Tang et al., 2025b). We also compare against a pre-trained fixed-length peptide SMILES MDM (Tang et al., 2025a), our pre-trained any-length MDM, and an A2D2 ablation without insertion and unmasking quality (**A2D2 w/o quality**). Additional details and ablations are in App E.

Results Fine-tuning any-length MDMs with our AJD weighted loss yields substantial reward gains across almost all objectives, with or without quality-based adaptive inference (App Fig. 2). At equal sampling steps, **A2D2** outperforms the any-length pre-trained baseline by a wide margin on most therapeutic properties, and surpasses both the fixed-length RL fine-tuning and the substantially more expensive multi-objective guidance baselines on *almost all objectives* (Table 2). Crucially, quality-based adaptive inference markedly improves sequence validity while optimizing the *same* rewards. This confirms that explicitly maximizing quality translates into more accurate, decodable generation.

5.3 Language Reasoning Experiment

Setup and Baselines We adapt the 8 billion parameter LLaDA-8B-Base (Nie et al., 2025) fixed-length MDM to an any-length MDM following Kim et al. (2026a). Then, we performed instruction fine-tuning (IFT) for math word problems with GSM8K (Cobbe et al., 2021) and code infilling with opc-sft-stage-2 (Huang et al., 2025). The IFT models are evaluated with GSM8K-test and HumanEval-infill (Bavarian et al., 2022) as the **Pretrained + IFT** baseline. Then, we additionally perform RL fine-tuning with A2D2. Additional experiment details are provided in App F.

Results Across both reasoning tasks, RL fine-tuning with **A2D2** yields large gains over the **Pre-trained + IFT** baseline at every matched inference budget (Table 3). On GSM8K, A2D2 raises Pass@1 from 35.71% to 60.96% at 128 sampling steps, a 25.3-point improvement, and the advantage persists at 256, 512, and 1024 steps. Notably, A2D2 reaches its best accuracy with only 128–256 steps and exceeds the baseline's strongest result (41.39% at 1024 steps) by roughly 20 points while using a fraction of the steps, demonstrating that reward alignment with A2D2 translates directly into greater inference efficiency. On HumanEval infilling, A2D2 again dominates the baseline at all budgets, improving exact match from 44.14% to 49.37% at 128 steps and from 48.89% to 57.12% at 1024 steps. Together, these results show that A2D2 simultaneously improves task accuracy and decoding efficiency for any-length language reasoning.

Table 3: **Reasoning experiment results.** GSM8k performance is evaluated with Pass@1 (%) out of 1319 total questions. Code infilling performance is evaluated with exact match to ground truth code (%) out of 1033 total examples.

	GSM8K			
Sampling Steps	**128**	**256**	**512**	**1024**
Pre-trained + IFT	35.71	37.98	39.88	41.39
A2D2	60.96	61.03	57.54	57.62
	HumanEval In-Filling			
Sampling Steps	**128**	**256**	**512**	**1024**
Pre-trained + IFT	44.14	44.82	47.53	48.89
A2D2	49.37	53.44	53.73	57.12

6 Conclusion

In this work, we introduce Fine-Tuning **A**ny-Length **D**iscrete **D**iffusion for **A**daptive Decoding (**A2D2**), a novel framework that unlocks reward-guided fine-tuning for any-length discrete diffusion models with adaptive quality-based inference. Driven by the importance of the inference schedule in any-length decoding, we define unmasking and insertion quality, establish their theoretical connection to compounding parallelization error and reconstruction likelihood, and co-optimize the policy and quality predictors to provably generate the optimal any-length path measure that samples from an intractable reward-tilted distribution. Our approach is a significant step toward any-length generation for diverse reward-based tasks.

AI USE STATEMENT

Generative AI tools were used for manuscript language editing and for assistance with code drafting and debugging. All reported numerical results were produced by the authors' computational pipelines and checked against saved outputs. The authors reviewed and revised all AI-assisted text and code and take responsibility for the final manuscript, analyses, and artifacts.

REPRODUCIBILITY STATEMENT

We will make the code publicly available on GitHub and provide installation scripts with documentation to address the dependency issues. Furthermore, all the datasets used in this paper are publicly available, with the experimental details provided in the Appendix. We hope that this will support and accelerate future research and development.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our research focuses on computational methods for language and biological sequence design. We acknowledge that biological sequence design technologies could have dual-use implications. The peptide design application demonstrated in our work targets therapeutic proteins for treating diseases, representing beneficial applications. However, we recognize that sequence design methods could potentially be misused for harmful purposes. We have designed our experiments to focus on clearly beneficial applications and have not explored potentially harmful uses. All datasets used in this work are publicly available and have been previously published with appropriate ethical considerations. No human subjects were involved in this research. The computational methods developed here are general frameworks that require domain expertise and appropriate oversight for real-world applications. We encourage responsible use of these methods and recommend that practitioners consider ethical implications and potential risks when applying sequence design technologies in practice.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. *International Conference on Learning Representations*.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Universal guidance for diffusion models. In *International Conference on Learning Representations*.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. 2012. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98.

Hanqun Cao, Haosen Shi, Chenyu Wang, Sinno Pan, and Pheng-Ann Heng. 2025. Glid^{2}e: A gradient-free lightweight fine-tune approach for discrete biological sequence design. *Advances in Neural Information Processing Systems*.

Jon Chambers, Mark Davies, Anna Gaulton, Anne Hersey, Sameer Velankar, Robert Petryszak, Janna Hastings, Louisa Bellis, Shaun McGlinchey, and John P Overington. 2013. Unichem: a unified chemical structure cross-referencing and identifier tracking system. *Journal of cheminformatics*, 5(1):3.

Sourav Chatterjee and Persi Diaconis. 2018. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135.

Haoxuan Chen, Yinuo Ren, Martin Renqiang Min, Lexing Ying, and Zachary Izzo. 2026. Solving inverse problems via diffusion-based priors: An approximation-free ensemble sampling approach. *Transactions on Machine Learning Research*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jacob K Christopher, Brian R Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. 2025. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12042–12059.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. 2025. Provable maximum entropy manifold exploration via diffusion models. *International Conference on Machine Learning*.

Fergal J Duffy, Mélanie Verniere, Marc Devocelle, Elise Bernard, Denis C Shields, and Anthony J Chubb. 2011. Cyclops: generating virtual libraries of cyclized and constrained peptides including nonnatural amino acids. *Journal of chemical information and modeling*, 51(4):829–836.

Peter Ertl and Ansgar Schuffenhauer. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8.

Aaron L Feller and Claus O Wilke. 2025. Peptide-aware chemical language model successfully predicts membrane diffusion of cyclic peptides. *Journal of Chemical Information and Modeling*, 65(2):571–579.

Nic Fishman, Leo Klarner, Valentin De Bortoli, Emile Mathieu, and Michael Hutchinson. 2023. Diffusion models for constrained domains. *Transactions on Machine Learning Research*.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. https://skylion007.github.io/OpenWebTextCorpus.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2026. Diffucoder: Understanding and improving masked diffusion models for code generation. *International Conference on Learning Representations*.

Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. 2023. Protein design with guided discrete diffusion. *Advances in Neural Information Processing Systems*.

Wei Guo, Yuchen Zhu, Molei Tao, and Yongxin Chen. 2024. Plug-and-play controllable generation for discrete masked models. *arXiv preprint arXiv:2410.02143*.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. 2025. Edit flows: Flow matching with edit operations. *Advances in Neural Information Processing Systems*.

Peter Holderrieth, Michael S Albergo, and Tommi Jaakkola. 2025. Leaps: A discrete neural sampler via locally equivariant networks. *International Conference on Machine Learning*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, et al. 2025. Opencoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 33167–33193.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. 2012. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2020. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pages 4849–4859. PMLR.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. 2026a. Any-order flexible length masked diffusion. *International Conference on Learning Representations*.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. 2026b. Fine-tuning masked diffusion for provable self-correction. *International Conference on Machine Learning*.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. 2025. Genmol: A drug discovery generalist with discrete diffusion. *International Conference on Machine Learning*.

Jianan Li, Keisuke Yanagisawa, Masatake Sugita, Takuya Fujie, Masahito Ohue, and Yutaka Akiyama. 2023. Cycpeptmpdb: a comprehensive database of membrane permeability of cyclic peptides. *Journal of Chemical Information and Modeling*, 63(7):2240–2250.

Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scalia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. 2025. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*.

Yanyan Li, Honghong Zhou, Xiaomin Chen, Yu Zheng, Quan Kang, Di Hao, Lili Zhang, Tingrui Song, Huaxia Luo, Yajing Hao, et al. 2021. Smprot: a reliable repository with comprehensive annotation of small proteins identified from ribosome profiling. *Genomics, proteomics & bioinformatics*, 19(4):602–610.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. 2025. Think while you generate: Discrete diffusion with planned denoising. *International Conference on Learning Representations*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. *International Conference on Machine Learning*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *Advances in Neural Information Processing Systems*.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. 2025. Unlocking guidance for discrete state-space diffusion and flow models. *International Conference on Learning Representations*.

Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan SC Lim, and Prudencio Tossou. 2024. Gotta be safe: a new framework for molecular design. *Digital Discovery*, 3(4):796–804.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *International Conference on Learning Representations*.

Yong-Hyun Park, Chieh-Hsin Lai, Satoshi Hayakawa, Yuhta Takida, and Yuki Mitsufuji. 2025. Jump your steps: Optimizing sampling schedule of discrete diffusion models. In *International Conference on Learning Representations*.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. 2025. Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Anru R Zhang, Michael Bronstein, Avishek Joey Bose, and Alexander Tong. 2026. Planner aware path learning in diffusion language models training. *International Conference on Learning Representations*.

Vignav Ramesh and Morteza Mardani. 2025. Test-time scaling of diffusion models via noise trajectory search. *Advances in Neural Information Processing Systems*.

Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Cheng-Hao Liu, Sarthak Mittal, Nouha Dziri, Michael M. Bronstein, Pranam Chatterjee, Alexander Tong, and Joey Bose. 2025. Steering masked discrete diffusion models via discrete denoising posterior prediction. In *International Conference on Learning Representations*.

Yinuo Ren, Haoxuan Chen, Yuchen Zhu, Wei Guo, Yongxin Chen, Grant M Rotskoff, Molei Tao, and Lexing Ying. 2025. Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms. *Advances in Neural Information Processing Systems*.

Kevin Rojas, Ye He, Chieh-Hsin Lai, Yuta Takida, Yuki Mitsufuji, and Molei Tao. 2026. Theory-informed improvements to classifier-free guidance for discrete diffusion models. *International Conference on Learning Representations*.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. 2025. Simple guidance mechanisms for discrete diffusion models. *International Conference on Learning Representations*.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. 2025. A general framework for inference-time scaling and steering of diffusion models. *International Conference on Machine Learning*.

Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alán Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. 2025. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. *International Conference on Machine Learning*.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*.

Sophia Tang, Yinuo Zhang, and Pranam Chatterjee. 2025a. Peptune: De novo generation of therapeutic peptides with multi-objective-guided discrete diffusion. *International Conference on Machine Learning*.

Sophia Tang, Yuchen Zhu, Molei Tao, and Pranam Chatterjee. 2025b. Tr2-d2: Tree search guided trajectory-aware fine-tuning for discrete diffusion. *arXiv preprint arXiv:2509.25171*.

Runchu Tian, Junxia Cui, Xueqiang Xu, Feng Yao, and Jingbo Shang. 2025. Finish first, perfect later: Test-time token-level cross-validation for diffusion large language models. *arXiv preprint arXiv:2510.05090*.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. 2024. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Hanchen Wang, and Tommaso Biancalani. 2025a. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. In *International Conference on Learning Representations*.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025b. Remasking discrete diffusion models with inference-time scaling. *Advances in Neural Information Processing Systems*.

Xinyou Wang, Zaixiang Zheng, Fei YE, Dongyu Xue, Shujian Huang, and Quanquan Gu. 2024. Diffusion language models are versatile protein learners. In *International Conference on Machine Learning*.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. 2025c. Dplm-2: A multimodal diffusion protein language model. *International Conference on Learning Representations*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *International Conference on Learning Representations*.

Oussama Zekri and Nicolas Boullé. 2025. Fine-tuning discrete diffusion models with policy gradient methods. *Advances in Neural Information Processing Systems*.

Yinuo Zhang, Sophia Tang, Tong Chen, Elizabeth Mahood, Sophia Vincoff, and Pranam Chatterjee. 2026. Peptiverse: A unified platform for therapeutic peptide property prediction. *Nature Communications*, 17:6819.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025a. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. 2025b. Informed correctors for discrete diffusion models. *Advances in Neural Information Processing Systems*.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. 2025. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *International Conference on Learning Representations*.

Renping Zhou, Zanlin Ni, Tianyi Chen, Zeyu Liu, Yang Yue, Yulin Wang, Yuxuan Wang, Jingshu Liu, and Gao Huang. 2025. Co-grpo: Co-optimized group relative policy optimization for masked diffusion model. *arXiv preprint arXiv:2512.22288*.

Yuchen Zhu, Wei Guo, Jaemoo Choi, Guan-Horng Liu, Yongxin Chen, and Molei Tao. 2025. Mdns: Masked diffusion neural sampler via stochastic optimal control. *Advances in Neural Information Processing Systems*.

Yuchen Zhu, Wei Guo, Jaemoo Choi, Petr Molodyk, Bo Yuan, Molei Tao, and Yongxin Chen. 2026. Enhancing reasoning for diffusion llms via distribution matching policy optimization. *International Conference on Machine Learning*.

A Related Work **17**

B Extended Theoretical Background **17**

B.1 Notation 17

B.2 Any-Length Masked Diffusion 17

C Theoretical Proofs **18**

C.1 Observations 18

C.2 Proof of Proposition 3.1 19

C.3 Proof of Proposition 3.3 20

C.4 Proof of Proposition 3.2 21

C.5 Proof of Proposition 3.4 22

C.6 Compounding Parallelization Error for Insertion and Unmasking 23

C.7 Radon-Nikodym Derivative Between Joint CTMCs 24

C.8 Adaptive Joint Decoding Loss 27

D Small Molecule Experiment Details **34**

D.1 Pre-Training the Any-Length Small-Molecule MDM 34

D.2 Rewards and Evaluation Metrics 34

D.3 Fine-Tuning Details 35

E Peptide Experiment Details **35**

E.1 Pre-Training the Any-Length Peptide MDM 35

E.2 Rewards and Evaluation Metrics 36

E.3 Fine-Tuning Details 36

F Language Reasoning Experiment Details **36**

F.1 Pre-Training the Any-Length Language MDM 36

F.2 Reasoning Tasks 37

F.3 Instruction Fine-Tuning 37

F.4 Reinforcement Learning Fine-Tuning with A2D2 37

F.5 Datasets and Rewards 38

G Ablation Studies **39**

H Additional Baselines 39

I Algorithms 44

J Example Generations 47

A Related Work

Inference-Time Scaling of Discrete Diffusion Inference-time scaling of diffusion models seeks to maximize the capabilities of pre-trained diffusion models for specialized tasks during inference, such as sampling from a reward-tilted distribution (Skreta et al., 2025; Singhal et al., 2025; Chen et al., 2026), constrained sampling (Fishman et al., 2023), or entropy maximization (De Santi et al., 2025). Specifically in the discrete state space, search-based methods (Tang et al., 2025a;b; Li et al., 2025; Ramesh and Mardani, 2025), importance-weighting techniques (Chatterjee and Diaconis, 2018), reward-gradient methods (Song et al., 2021; Bansal et al., 2024), and classifier-based and classifier-free guidance (Nisonoff et al., 2025; Rector-Brooks et al., 2025; Wang et al., 2024; Schiff et al., 2025; Rojas et al., 2026; Guo et al., 2024) have been explored.

Training-Time Fine-Tuning of Discrete Diffusion Training-time fine-tuning methods instead update the model parameters to optimize a downstream reward. One line of work backpropagates rewards through the denoising trajectory, as in DRAKES (Wang et al., 2025a), which makes the otherwise non-differentiable discrete trajectories differentiable via the Gumbel-Softmax trick to optimize sequences that are both high-reward and natural-like. A complementary line adapts policy-gradient reinforcement learning to discrete diffusion: SEPO (Zekri and Boullé, 2025) provides a framework that handles non-differentiable rewards via self-normalized importance sampling, while a growing body of work adapts GRPO to masked diffusion language models, including diffu-GRPO (Zhao et al., 2025a) and Coupled GRPO (Gong et al., 2026). TR2-D2 (Tang et al., 2025b) instead casts fine-tuning as stochastic optimal control, using Monte Carlo tree search to curate replay buffers of high-reward trajectories for off-policy RL, and extends to multi-objective reward optimization.

Optimization of Inference Schedule for Discrete Diffusion Closest to our work is PRISM (Kim et al., 2026b), which introduces *per-token quality* as a learned measure that determines whether to re-mask already unmasked tokens during inference. However, it remains limited to fixed-length masked diffusion for optimizing sampling of the data distribution. Co-GRPO (Zhou et al., 2025) introduces a reward fine-tuning strategy for fixed-length masked diffusion that simultaneously optimizes the inference schedule and policy model with a shared reward signal via group-relative policy optimization (GRPO). Jump Your Steps (JYS) (Park et al., 2025) optimizes the times at which discrete jumps are made during inference by minimizing a KL upper bound on the compounding decoding error for fixed-length masked diffusion models. Related inference-time mechanisms include *planner*-guided decoding, which learns a planner to choose which tokens to reveal at each step (Liu et al., 2025; Peng et al., 2025; 2026), as well as remasking and corrector-style methods that iteratively revise intermediate predictions to improve sample quality (Wang et al., 2025b; Zhao et al., 2025b; Tian et al., 2025).

B Extended Theoretical Background

B.1 Notation

We denote the CTMC of a diffusion trajectory as $\boldsymbol{X}_{0:1} := (\boldsymbol{X}_t)_{t\in[0,1]}$ which lies in the variable-length discrete state space $\mathcal{X} = \bigcup_l \mathcal{V}^l$, where $\mathcal{V} := \{1, \dots, V\}^l$ of sequences of length l and vocabulary size V, where l is the variable sequence length and L is the maximum sequence length. An any-length discrete diffusion model is characterized by a path measure $\mathbb{P} \in \mathcal{P}(C([0,T];\mathcal{X}))$ over trajectories that follow a joint generator $\boldsymbol{A}_t = \boldsymbol{Q}_t + \boldsymbol{R}_t$, where $\boldsymbol{Q}_t$ is the unmasking rate and $\boldsymbol{R}_t$ is the insertion rate. Specifically, we denote the pre-trained discrete diffusion model as $\mathbb{P}^{\text{pre}}$ and the optimal target model as $\mathbb{P}^\star$. For a corrupted sequence $\boldsymbol{x}_t$ at time t, the unmasking posterior is given by $f_\theta(\boldsymbol{x}, t)$, with indices $[\ell, \boldsymbol{v}]$ denoting the probability of a single token at position ℓ and the insertion rate is given by $g_\theta(\boldsymbol{x}_t, t)[\ell]$, with index $[\ell]$ indicating the insertion expectation between positions $\ell - 1$ and ℓ of the sequence.

B.2 Any-Length Masked Diffusion

In this work, we build upon the any-length masked diffusion model framework introduced in Kim et al. (2026a), which leverages a pair of joint stochastic interpolants for the insertion and unmasking

schedules. Here, we will provide a comprehensive background on the any-length MDM framework and highlight its distinction from standard fixed-length MDMs.

Forward Noising Process In standard MDM, the forward process involves applying mask tokens gradually across the sequence until the sequence is fully masked at time $t = 0$. For any-length MDMs, the forward process consists not only of the masking step but also of a **deletion** step, which serves as the inverse operation for the insertion step during inference. Therefore, we must define both the insertion time t_i^ℓ and the unmasking time t_u^ℓ for each token position $\boldsymbol{x}^\ell$ as:

$$t_i^\ell \sim \dot{\alpha}_t dt, \quad t_u^\ell \sim \mathbf{1}[t \geq t_i^\ell] \frac{\dot{\beta}}{1-\beta_{t_i^\ell}} dt, \quad \boldsymbol{x}_t^\ell = \begin{cases} (\text{empty}), & 0 < t < t_i^\ell \\ \boldsymbol{M}, & t_i^\ell \leq t \leq t_u^\ell \\ \boldsymbol{x}_1^{s_t[\ell]}, & t_u^\ell \leq t \leq 1 \end{cases} \tag{22}$$

where (empty) indicates a position that has not been inserted yet. We denote the index of a token in the partial subsequence with s_t given by:

$$s_t := \{\ell \in \{1, \dots, \text{len}(\boldsymbol{x}_1)\} |\ t_i^\ell \leq t\} \tag{23}$$

where $s_t[\ell]$ is the index of the token $\boldsymbol{x}_1^\ell$ in $\boldsymbol{x}_t$ at time t and we have $\text{len}(s_t) \leq \text{len}(\boldsymbol{x}_t)$.

Parameterization Just like fixed-length MDMs, the masking step in any-length MDMs has an inverse unmasking step that is parameterized with an **unmasking posterior** $f_\theta(\boldsymbol{x}_t, t)[\ell] : \mathcal{V}^L \times [0,1] \to \Delta^D$ that predicts the distribution over the token vocabulary for each masked position. This model is trained with the standard denoising cross-entropy loss defined as:

$$\mathcal{L}_{\text{unmask}}(\theta; \boldsymbol{x}_1) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)} \mathop{\mathbb{E}}_{\boldsymbol{x}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \left[-\frac{\dot{\beta}_t}{1-\beta_t} \sum_{\ell:\boldsymbol{x}_t^\ell = \boldsymbol{M}} \log f_\theta(\boldsymbol{x}_t, t)[\ell, \boldsymbol{x}_1^{s_t[\ell]}] \right] \tag{24}$$

Since any-length MDMs contain the additional deletion step, we parameterize the inverse **insertion** step with an **insertion expectation** $g_\theta(\boldsymbol{x}_t, t)[\ell] : \mathcal{V}^L \times [0,1] \to \mathbb{R}_{>0}$ which predicts the number of tokens that still need to be inserted between tokens $\boldsymbol{x}_t^{\ell-1}$ and $\boldsymbol{x}_t^\ell$. This model is trained by minimizing the scalar **Bregman divergence** $\phi(x, y) = y - x \log y$ between the predicted insertions and the true number of tokens that need to be inserted at each position:

$$\mathcal{L}_{\text{insert}}(\theta; \boldsymbol{x}_1) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)} \mathop{\mathbb{E}}_{\boldsymbol{x}_t\sim p_t(\cdot|\boldsymbol{x}_1)} \left[-\frac{\dot{\alpha}_t}{1-\alpha_t} \sum_{\ell=0}^{\text{len}(\boldsymbol{x}_t)} \phi\left(s_t[\ell] - s_t[\ell-1] - 1, g_\theta(\boldsymbol{x}_t, t)[\ell]\right) \right] \tag{25}$$

Minimizing both losses $\mathcal{L}_{\text{unmask}}(\theta)$ and $\mathcal{L}_{\text{insert}}(\theta)$ yields the *unique* optimal value, which defines the rate matrices of the reverse process given by:

$$\boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{x}^{\triangleleft_\ell \boldsymbol{M}}) = \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot \mathbb{E}[s_t[\ell] - s_t[\ell-1] - 1 |\ \boldsymbol{x}_t = \boldsymbol{x}] = \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot g_\theta(\boldsymbol{x}, t)[\ell] \qquad \text{(Insertion Rate)}$$

$$\boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{x}^{\ell \leftarrow \boldsymbol{v}}) = \frac{\dot{\beta}_t}{1-\beta_t} \cdot \mathbb{P}(\boldsymbol{x}_1^{s_t[\ell]} = \boldsymbol{v} | \boldsymbol{x}_t = \boldsymbol{x}) = \frac{\dot{\beta}_t}{1-\beta_t} \cdot f_\theta(\boldsymbol{x}, t)[\ell, \boldsymbol{v}] \qquad \text{(Unmasking Rate)}$$

Adaptive Sampling Guarantee A **crucial feature** of any-length MDMs is that the values of the rate matrix are *decoupled from the choice of masking schedule* β_t. This means that the unmasking posterior learned by $f_\theta(\boldsymbol{x}_t, t)$ and insertion expectation $g_\theta(\boldsymbol{x}_t, t)$ capture all possible unmasking transitions that interpolate between the empty prior p_0 and the target distribution p_{target}. Given this, Kim et al. (2026a) proves that any adaptive any-length MDM sampler for the p_{target} will generate a sequence of intermediate states $\{\boldsymbol{x}_{t_i}\}_{i=1}^{N_{\text{steps}}}$ where the final sequence $\boldsymbol{x}_1$ is a sample from p_{target}.

C Theoretical Proofs

C.1 Observations

Throughout the theoretical proofs, we make the following **key observations**:

(i) The unmasking rate $\boldsymbol{Q}_t$ and insertion rate $\boldsymbol{R}_t$ do not overlap, such that:

$$\boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{y}) = 0, \quad \forall t, \boldsymbol{x}, \boldsymbol{y} \tag{26}$$

which follows from the fact that insertion and unmasking actions are disjoint.

(ii) The total generator $\boldsymbol{A}_t$ decomposes additively:

$$\boldsymbol{A}_t(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{y}), \quad \forall t, \boldsymbol{x}, \boldsymbol{y} \tag{27}$$

(iii) Both insertion and unmasking rates are *valid* such that:

$$\boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{x}) = -\sum_{y \neq x} \boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{y}), \quad \sum_{y \neq x} \boldsymbol{Q}_t(\boldsymbol{x}, \boldsymbol{y}) < \infty \tag{28}$$

$$\boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{x}) = -\sum_{y \neq x} \boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{y}), \quad \sum_{y \neq x} \boldsymbol{R}_t(\boldsymbol{x}, \boldsymbol{y}) < \infty \tag{29}$$

which ensures that the CTMC is well-defined such that reward tilting preserves the generator structure.

(iv) The tilted generators $\boldsymbol{Q}_t^\star$ and $\boldsymbol{R}_t^\star$ and the reference generators $\boldsymbol{Q}_t^0$ and $\boldsymbol{R}_t^0$ have the same support, such that:

$$\boldsymbol{Q}_t^0(\boldsymbol{x}, \boldsymbol{y}) = 0 \implies \boldsymbol{Q}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = 0 \tag{30}$$

$$\boldsymbol{R}_t^0(\boldsymbol{x}, \boldsymbol{y}) = 0 \implies \boldsymbol{R}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = 0 \tag{31}$$

which ensures that the path measures $\mathbb{P}^0$ and $\mathbb{P}^\star$ are mutually absolutely continuous.

C.2 Proof of Proposition 3.1

Proposition 3.1 (Unique Minimizer of Unmasking Quality Loss)**.** *The unique minimizer of the unmasking quality loss $\mathcal{L}_{UQL}(\phi)$ is the true unmasking quality:*

$$\mu_\star = \mu_{\phi^\star}, \quad \textit{where} \quad \phi^\star = \arg\min_\phi \mathcal{L}_{UQL}(\phi) \tag{7}$$

Proof. First, we recall the *Unmasking Quality Loss* $\mathcal{L}_{\text{UQL}}$ defined in (6) as:

$$\mathcal{L}_{\text{UQL}}(\phi; \boldsymbol{x}_1) := \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)} \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_t\sim p_t(\cdot|\boldsymbol{x}_1), \boldsymbol{y}\sim p_{s|t}(\cdot|\tilde{\boldsymbol{x}}_t)} \left[\sum_{\ell \in \mathcal{M}} \text{BCE}\left(\mathbf{1}[\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}], \mu_\phi^\ell(\boldsymbol{y}) \right) \right] \tag{32}$$

Given a clean sequence $\boldsymbol{x}_1 \sim p_{\text{data}}$, let $\tilde{\boldsymbol{x}}_t$ be a sample from the posterior $\tilde{\boldsymbol{x}}_t \sim p_t(\cdot|\boldsymbol{x}_1)$. Then, we sample a subset $\mathcal{M}_t$ of masked indices in $\tilde{\boldsymbol{x}}_t$ to unmask to obtain $\boldsymbol{y}$ where $\forall \ell \in \mathcal{M}_t$, we sample $\boldsymbol{y}^\ell \sim f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell]$.

Fix a position $\ell \in \mathcal{M}_t$ and consider the random binary variable b_ℓ defined by categorically sampling from $f_\theta(\tilde{\boldsymbol{x}}_t, t)[\ell]$ and the model predicted token quality $c_\ell(\boldsymbol{y}) := \mu_\phi^\ell(\boldsymbol{y})$:

$$b^\ell := \mathbf{1}[\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}] \in \{0, 1\}, \quad c_\ell(\boldsymbol{y}) := \mu_\phi^\ell(\boldsymbol{y}) \in [0, 1] \tag{33}$$

For every position $\ell \in \mathcal{M}_t$, we minimize the following loss:

$$\mathcal{L}_{\text{UQL}}(\phi; \ell) := \mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t, \boldsymbol{y}} \left[\text{BCE}\left(b_\ell, c_\ell(\boldsymbol{y}) \right) \right] \tag{34}$$

Applying the law of total expectation, we have:

$$\mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t, \boldsymbol{y}} \left[\text{BCE}\left(b_\ell, c_\ell(\boldsymbol{y}) \right) \right] = \mathbb{E}_{\boldsymbol{y}} \underbrace{\left[\mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t} \left[\text{BCE}(b_\ell, c_\ell(\boldsymbol{y})) \middle| \boldsymbol{y} \right] \right]}_{:= D_\ell(\boldsymbol{y}, c_\ell)} \tag{35}$$

Let us define the conditional random variable:

$$q_\ell(\boldsymbol{y}) := p(b_\ell = 1|\boldsymbol{y}) = p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]}|\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t} \left[b_\ell | \boldsymbol{y} \right] \tag{36}$$

Then, we can write the BCE loss $D_\ell(\boldsymbol{y}, c)$ as:

$$D_\ell(\boldsymbol{y}, c_\ell) = -q_\ell(\boldsymbol{y}) \log c_\ell - (1 - q_\ell(\boldsymbol{y})) \log(1 - c_\ell) \tag{37}$$

Differentiating with respect to the parameters ϕ denoted as c_ℓ and setting the derivative equal to 0, we get that the minimizer is:

$$\frac{d}{dc_\ell} D_\ell(\boldsymbol{y}, c_\ell) = -\frac{q_\ell(\boldsymbol{y})}{c_\ell} + \frac{1 - q_\ell(\boldsymbol{y})}{1 - c_\ell} \implies \frac{q_\ell(\boldsymbol{y})}{c_\ell^\star} = \frac{1 - q_\ell(\boldsymbol{y})}{1 - c_\ell^\star} \implies c_\ell^\star = q_\ell(\boldsymbol{y}) \tag{38}$$

Taking the second derivative with respect to c_ℓ, we show that this minimizer is unique:

$$\frac{d^2}{dc_\ell^2} D_\ell(\boldsymbol{y}, c_\ell) = \frac{q_\ell(\boldsymbol{y})}{c_\ell^2} + \frac{1 - q_\ell(\boldsymbol{y})}{(1 - c_\ell)^2} \geq 0 \tag{39}$$

Since we defined the true unmasking quality as $\mu_\star^\ell(\boldsymbol{y}) := p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^{s_t[\ell]} | \boldsymbol{y})$, the minimizer $c^\star$ exactly matches the true unmasking posterior:

$$c_\ell^\star = q_\ell(\boldsymbol{y}) = \mu_\star^\ell(\boldsymbol{y}) = \arg\min_{c_\ell \in [0,1]} \{D_\ell(\boldsymbol{y}, c_\ell)\} \tag{40}$$

Since for all pairs $(\boldsymbol{y}, \ell)$, we have shown that the BCE loss $D_\ell(\boldsymbol{y}, c_\ell)$ is minimized at $\mu_\phi^\ell(\boldsymbol{y}) = \mu_\star^\ell(\boldsymbol{y})$, so the function that minimizes the full objective $\mathcal{L}_{\text{UQL}}(\phi) := \mathbb{E}_{\boldsymbol{y}}[D_\ell(\boldsymbol{y}, c_\ell)]$ must be the true function $\mu_\star$. □

C.3 Proof of Proposition 3.3

Proposition 3.3 (Minimizer of Insertion Quality Loss)**.** *The unique minimizer of the insertion quality loss $\mathcal{L}_{IQL}(\phi)$ is the true insertion quality:*

$$\nu_\star = \nu_{\phi^\star}, \quad \textit{where} \quad \phi^\star = \arg\min_\phi \mathcal{L}_{IQL}(\phi) \tag{14}$$

Proof. First, we recall the *Insertion Quality Loss* $\mathcal{L}_{\text{IQL}}$ defined in (13) as:

$$\mathcal{L}_{\text{IQL}}(\phi) := \mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t, \boldsymbol{y}} \left[\sum_{i \in \mathcal{I}} \text{BCE}\big(\nu_\star^i(\boldsymbol{y}), \nu_\phi^i(\boldsymbol{y})\big) \right] \tag{41}$$

where the BCE is defined as:

$$\text{BCE}(b, c) = -b \log c - (1 - b) \log(1 - c), \quad b, c \in [0, 1] \tag{42}$$

Since the sum $\sum_{i \in \mathcal{I}} \text{BCE}\big(\nu_\star^i(\boldsymbol{y}), \nu_\phi^i(\boldsymbol{y})\big)$ is inside the expectation, and the loss is calculated independently for each position i given $\nu_\phi^i(\boldsymbol{y})$, minimizing $\mathcal{L}_{\text{IQL}}(\phi)$ is equivalent to minimizing the following for every inserted position i:

$$\mathcal{L}_{\text{IQL}}(\phi; i) := \mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t, \boldsymbol{y}} \left[\text{BCE}\big(\nu_\star^i(\boldsymbol{y}), \nu_\phi^i(\boldsymbol{y})\big) \right] \tag{43}$$

Applying the law of total expectation, we have:

$$\mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t, \boldsymbol{y}} \left[\text{BCE}\big(\nu_\star^i(\boldsymbol{y}), \nu_\phi^i(\boldsymbol{y})\big) \right] = \mathbb{E}_{\boldsymbol{y}} \underbrace{\left[\mathbb{E}_{\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_t} \left[\text{BCE}\big(\nu_\star^i(\boldsymbol{y}), \nu_\phi^i(\boldsymbol{y})\big) \middle| \boldsymbol{y} \right] \right]}_{:= B_i(\boldsymbol{y}, c)} \tag{44}$$

where we define $c_i := \nu_\phi^i(\boldsymbol{y}) \in [0, 1]$ as the scalar predicted insertion quality at position i and $B_i(\boldsymbol{y}, c_i)$ as the prediction error given the input sequence $\boldsymbol{y}$ and predicted quality c_i. Then, it suffices to prove that for all $(\boldsymbol{y}, i)$, the prediction error is *uniquely minimized* at $c_i = \nu_\star^i(\boldsymbol{y})$.

Fixing a pair $(\boldsymbol{y}, i)$, we can define:

$$b_i := \nu_\star^i(\boldsymbol{y}) \in [0, 1], \quad c_i := \nu_\phi^i(\boldsymbol{y}) \in [0, 1] \tag{45}$$

which yields the prediction error objective:

$$B_i(\boldsymbol{y}, c_i) = -b_i \log c_i - (1 - b_i) \log(1 - c_i) \tag{46}$$

Differentiating with respect to c_i (parameterized model), we have:

$$\frac{d}{dc_i} B_i(\boldsymbol{y}, c_i) = -\frac{b_i}{c_i} + \frac{1-b_i}{1-c_i} \tag{47}$$

and setting the derivative to zero, we get the minimizer:

$$-\frac{b_i}{c_i^\star} + \frac{1-b_i}{1-c_i^\star} = 0 \implies \frac{b_i}{c_i^\star} = \frac{1-b_i}{1-c_i^\star} \implies c_i^\star = b_i \tag{48}$$

To prove uniqueness, we take the second derivative:

$$\frac{d^2}{dc_i^2} B_i(\boldsymbol{y}, c_i) = \frac{b_i}{c_i^2} + \frac{1-b_i}{(1-c_i)^2} \geq 0 \tag{49}$$

So, the objective function is convex in c_i and the minimizer is unique. Therefore, we conclude that:

$$b_i = \nu_\star^i(\boldsymbol{y}) = \arg\min_{c_i \in [0,1]} \{B_i(\boldsymbol{y}, c_i)\} \tag{50}$$

We have shown that for all $(\boldsymbol{y}, i)$, the prediction error is uniquely minimized at $\nu_\phi^i(\boldsymbol{y}) = \nu_\star^i(\boldsymbol{y})$, the only function ν_ϕ that minimizes the outer expectation $\mathbb{E}_{\boldsymbol{y}}[B_i(\boldsymbol{y}, c_i)]$ is the true function $\nu_\star$. □

C.4 Proof of Proposition 3.2

Proposition C.1 (Optimal Parallel Unmasking as Maximizing the Unmasking Quality)**.** *Assume that in a parallel unmasking step on indices $\mathcal{M}_t = \{\ell_k\}_{k=1}^K$, the unmasked tokens are conditionally independent given the unchanged context $\bar{\boldsymbol{x}}_s$ and the current state $\boldsymbol{x}_s$, i.e. $p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^K p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s)$. If we define the unmasking quality $\mu_\star^{\ell_k}(\boldsymbol{x}_t) := p(\boldsymbol{x}_t^{\ell_k} = \boldsymbol{x}_1^{\ell_k} | \boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}})$ and note that $p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = p(\boldsymbol{x}_t^{\ell_k} | \boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}})$, then*

$$p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^K \mu_\star^{\ell_k}(\boldsymbol{x}_t) \tag{51}$$

Furthermore, the compounding parallelization error can be written as the KL divergence

$$\mathcal{E}_{\text{unmask}}^{\text{CPE}}(s \to t | \boldsymbol{x}_s) := D_{\text{KL}}\left(p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) \,\middle\|\, \prod_{k=1}^K p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) \right) \geq 0 \tag{52}$$

and admits the decomposition

$$\mathcal{E}_{\text{unmask}}^{\text{CPE}}(s \to t | \boldsymbol{x}_s) = \mathbb{E}_{p_t}\left[\log p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) \right] - \sum_{k=1}^K \mathbb{E}_{p_t}\left[\log \mu_\star^{\ell_k}(\boldsymbol{x}_t) \right] \tag{53}$$

with equality to 0 if and only if the KL divergence is 0 (i.e., optimal parallel decoding).

Proof. First, we recall the definition of token quality. Given a partially masked and deleted sequence $\boldsymbol{x}_t$, the unmasking quality of each token $\boldsymbol{x}_t^\ell$ is the probability that given the sequence masked at position ℓ, denoted $\tilde{\boldsymbol{x}}_s := \boldsymbol{x}_t^{\ell \leftarrow \boldsymbol{M}}$, we sample $\tilde{\boldsymbol{x}}_s^\ell = \boldsymbol{x}_t^\ell$ via the unmasking posterior $f_\theta(\tilde{\boldsymbol{x}}_s, s)$. This is written as:

$$\mu_\star^\ell(\tilde{\boldsymbol{x}}_s) = p(\tilde{\boldsymbol{x}}_s^\ell = \boldsymbol{x}_t^\ell | \boldsymbol{x}_t^{\ell \leftarrow \boldsymbol{M}}) \tag{54}$$

Consider a unmasking step $\boldsymbol{x}_s \to \boldsymbol{x}_t$ where a subset of K tokens with indices $\mathcal{M}_t := \{\ell_k\}_{k=1}^K$ are unmasked where $\bar{\boldsymbol{x}}_s$ is the subset of the sequence that remains unchanged from $s \to t$. Let the joint conditional and the product of marginals conditioned on $\boldsymbol{x}_s$ be denoted as:

$$p_{\mathcal{M}_t}(\cdot) := p(\boldsymbol{x}_t^{\mathcal{M}_t} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s), \quad q_{\mathcal{M}_t}(\cdot) := \prod_{k=1}^K p(\boldsymbol{x}_t^{\ell_k} | \bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) \tag{55}$$

Then, the compounding parallelization error (CPE) of the step is given by:

$$\mathcal{E}^{\text{CPE}}_{\text{unmask}}(s \to t|\boldsymbol{x}_s) = D_{\text{KL}}\left(p_{\mathcal{M}_t} \| q_{\mathcal{M}_t}\right) \geq 0 \tag{56}$$

where $\mathcal{E}^{\text{CPE}}_{\text{unmask}}(s \to t|\boldsymbol{x}_s) = 0$ if and only if $p_{\mathcal{M}_t} = q_{\mathcal{M}_t}$ and all the unmasked tokens are *conditionally independent* such that the product of the marginal probabilities and the joint probability is equal given $(\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s)$. Expanding the KL divergence yields:

$$\mathcal{E}^{\text{CPE}}_{\text{unmask}}(s \to t|\boldsymbol{x}_s) = \mathbb{E}_{\boldsymbol{x}_t^{\mathcal{M}_t} \sim p_{\mathcal{M}_t}}\left[\log p(\boldsymbol{x}_t^{\mathcal{M}_t}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) - \sum_{k=1}^{K} \log p(\boldsymbol{x}_t^{\ell_k}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s)\right] \geq 0 \tag{57}$$

In the case where all unmasked tokens are conditionally independent, we have:

$$p(\boldsymbol{x}_t^{\mathcal{M}_t}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^{K} p(\boldsymbol{x}_t^{\ell_k}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) \tag{58}$$

In addition, under conditional independence of $\mathcal{M}_t$, the probability of a particular unmasked token $\boldsymbol{x}_t^{\ell_k}$ should be the same given the context $\boldsymbol{x}_s$ and given the context with only ℓ_k masked, $\boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}}$, which aligns with the definition of the true unmasking quality. Therefore,

$$p(\boldsymbol{x}_t^{\ell_k}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = p(\boldsymbol{x}_t^{\ell_k}|\boldsymbol{x}_t^{\ell_k \leftarrow \boldsymbol{M}}) =: \mu_\star^\ell(\boldsymbol{x}_t) \tag{59}$$

Substituting (59) into the optimal parallel unmasking condition in (58), we have:

$$p(\boldsymbol{x}_t^{\mathcal{M}_t}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^{K} p(\boldsymbol{x}_t^{\ell_k}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) = \prod_{k=1}^{K} \mu_\star^\ell(\boldsymbol{x}_t) \tag{60}$$

which means that maximizing the product of token qualities maximizes the probability of optimal parallel unmasking. We can also derive a tractable Monte Carlo estimator for the CPE in (57) as:

$$\widehat{\mathcal{E}}^{\text{CPE}}_{\text{unmask}}(s \to t|\boldsymbol{x}_s) = \log p(\boldsymbol{x}_t^{\mathcal{M}_t}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s) - \sum_{k=1}^{K} \log p(\boldsymbol{x}_t^{\ell_k}|\bar{\boldsymbol{x}}_s, \boldsymbol{x}_s), \quad \mathcal{E}^{\text{CPE}}_{\text{unmask}} = \mathbb{E}_{p_t}\left[\widehat{\mathcal{E}}^{\text{CPE}}_{\text{unmask}}\right] \geq 0 \tag{61}$$

which concludes our proof. □

C.5 Proof of Proposition 3.4

Proposition 3.4 (Insertion Quality as an Upper Bound on Reconstruction via Insertions)**.** *Consider a clean target sequence $\boldsymbol{x}_1 \sim p_{target}$ and an intermediate sequence $\boldsymbol{x}_s \sim p_s(\cdot|\boldsymbol{x}_1)$. Given a set of indices of inserted masks at each gap $\mathcal{I}_t$ which yields $\boldsymbol{x}_t^{\mathcal{I}_t}$, the probability of reconstructing $\boldsymbol{x}_1$ is upper-bounded by the product of insertion qualities:*

$$p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t}|\boldsymbol{x}_t) \leq \prod_{i \in \mathcal{I}_t} \nu_\star^i(\boldsymbol{x}_t) \approx \prod_{i \in \mathcal{I}_t} \nu_\phi^i(\boldsymbol{x}_t) \tag{15}$$

where the unmasking posterior for each inserted mask is conditionally independent given $\boldsymbol{x}_t$.

Proof. First, we recall the definition of insertion quality as the sum of the probabilities of all possible tokens that exist within the gap in the ground truth sequence, defined as:

$$\nu_\star(\boldsymbol{y}) := \max_{s_t[\ell-1] < i < s_t[\ell]} \left\{p(\boldsymbol{y}^\ell = \boldsymbol{x}_1^i|\boldsymbol{y})\right\} \tag{62}$$

Consider a target sequence $\boldsymbol{x}_1 \sim p_{\text{target}}$. After an insertion step $(\mathcal{I}_t, \boldsymbol{x}_t) \sim g_\theta(\boldsymbol{x}_s, s)$, we unmask the masked tokens to reconstruct $\boldsymbol{x}_1$. For this proof, we assume that the unmasking posterior factorizes across all masked tokens, which gives the equality:

$$p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t}|\boldsymbol{x}_t) = \prod_{i \in \mathcal{I}} p(\boldsymbol{x}_t^i = \boldsymbol{x}_1^i|\boldsymbol{x}_t) \tag{63}$$

$$\log p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t}|\boldsymbol{x}_t) = \sum_{i \in \mathcal{I}} \log p(\boldsymbol{x}_t^i = \boldsymbol{x}_1^i|\boldsymbol{x}_t) \tag{64}$$

Since we define the **insertion quality** $\nu_\star^i$ at an inserted mask position i as the largest probability mass assigned by the unmasking posterior to any of the ground truth tokens at the gap $\mathcal{G}_i := [s_t[i-1], s_t[i]]$, we can write:

$$p(\boldsymbol{x}_t^i = \boldsymbol{x}_1^{\mathcal{G}_i} | \boldsymbol{x}_t) \leq \nu_\star^i(\boldsymbol{x}_t) \implies \log p(\boldsymbol{x}_t^i = \boldsymbol{x}_1^{\mathcal{G}_i} | \boldsymbol{x}_t) \leq \log \nu_\star^i(\boldsymbol{x}_t) \tag{65}$$

Assuming conditional independence across all inserted masks given $\boldsymbol{x}_t$,we get:

$$p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t} | \boldsymbol{x}_t) = \prod_{i \in \mathcal{I}} p(\boldsymbol{x}_t^i = \boldsymbol{x}_1^{\mathcal{G}_i} | \boldsymbol{x}_t) \leq \prod_{i \in \mathcal{I}} \nu_\star^i(\boldsymbol{x}_t) \tag{66}$$

where $\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t}$ means that target tokens are aligned to inserted slots. Taking the expectation over $(\mathcal{I}_t, \boldsymbol{x}_t) \sim p(\mathcal{I}_t, \boldsymbol{x}_t | \boldsymbol{x}_s)$, we have:

$$\mathbb{E}_{p(\mathcal{I}_t, \boldsymbol{x}_t | \boldsymbol{x}_s)} \left[p(\boldsymbol{x}_t^{\mathcal{I}_t} = \boldsymbol{x}_1^{\mathcal{I}_t} | \boldsymbol{x}_t) \right] \leq \mathbb{E}_{p(\mathcal{I}_t, \boldsymbol{x}_t | \boldsymbol{x}_s)} \left[\prod_{i \in \mathcal{I}} \nu_\star^i(\boldsymbol{x}_t) \right] \tag{67}$$

Therefore, maximizing $\prod_{i \in \mathcal{I}} \nu_\star^i(\boldsymbol{x}_t)$ maximizes a tractable upper bound on the probability that all inserted slots are reconstructed correctly, providing a principled surrogate objective for insertion scheduling. □

C.6 Compounding Parallelization Error for Insertion and Unmasking

First, we establish the following lemma that decomposes the KL divergence between joint distributions.

Lemma C.1 (KL Divergence Chain Rule)**.** *The KL divergence between the joint distributions $\mathbb{P}_{s,t}(\boldsymbol{X}_s, \boldsymbol{X}_t)$ and $\mathbb{P}'_{s,t}(\boldsymbol{X}_s, \boldsymbol{X}_t)$ on the joint space $\mathcal{X} \times \mathcal{X}$ where $\mathcal{X}$ is a finite discrete state space, satisfies:*

$$D_{KL}(\mathbb{P}_{s,t} \| \mathbb{P}'_{s,t}) = D_{KL}(\mathbb{P}_s \| \mathbb{P}'_s) + \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s} \left[D_{KL}(\mathbb{P}_{t|s}(\cdot|\boldsymbol{x}_s) \| \mathbb{P}'_{t|s}(\cdot|\boldsymbol{x}_s) \right] \tag{68}$$

where $\mathbb{P}_s, \mathbb{P}'_s$ denote marginal distributions and $\mathbb{P}_{t|s}, \mathbb{P}'_{t|s}$ denote conditional distributions.

Proof. By expanding the definition of KL divergence, we have:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{P}_{s,t} \| \mathbb{P}'_{s,t}) &= \sum_{\boldsymbol{x}_s, \boldsymbol{x}_t} \mathbb{P}_{s,t}(\boldsymbol{x}_s, \boldsymbol{x}_t) \log \frac{\mathbb{P}_{s,t}(\boldsymbol{x}_s, \boldsymbol{x}_t)}{\mathbb{P}'_{s,t}(\boldsymbol{x}_s, \boldsymbol{x}_t)} \\
&= \sum_{\boldsymbol{x}_s, \boldsymbol{x}_t} \mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s) \mathbb{P}_s(\boldsymbol{x}_s) \log \frac{\mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s) \mathbb{P}_s(\boldsymbol{x}_s)}{\mathbb{P}'_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s) \mathbb{P}'_s(\boldsymbol{x}_s)} \\
&= \underbrace{\sum_{\boldsymbol{x}_s, \boldsymbol{x}_t} \mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)}_{=1} \mathbb{P}_s(\boldsymbol{x}_s) \log \frac{\mathbb{P}_s(\boldsymbol{x}_s)}{\mathbb{P}'_s(\boldsymbol{x}_s)} + \sum_{\boldsymbol{x}_s, \boldsymbol{x}_t} \mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s) \mathbb{P}_s(\boldsymbol{x}_s) \log \frac{\mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)}{\mathbb{P}'_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)} \\
&= \underbrace{\sum_{\boldsymbol{x}_s} \mathbb{P}_s(\boldsymbol{x}_s) \log \frac{\mathbb{P}_s(\boldsymbol{x}_s)}{\mathbb{P}'_s(\boldsymbol{x}_s)}}_{D_{\mathrm{KL}}(\mathbb{P}_s \| \mathbb{P}_s)} + \sum_{\boldsymbol{x}_s} \mathbb{P}_s(\boldsymbol{x}_s) \underbrace{\sum_{\boldsymbol{x}_s, \boldsymbol{x}_t} \mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s) \log \frac{\mathbb{P}_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)}{\mathbb{P}'_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)}}_{D_{\mathrm{KL}}(\mathbb{P}_{t|s} \| \mathbb{P}_{t|s})} \\
&= D_{\mathrm{KL}}(\mathbb{P}_s \| \mathbb{P}_s) + \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s}[D_{\mathrm{KL}}(\mathbb{P}_{t|s} \| \mathbb{P}_{t|s})]
\end{aligned} \tag{69}$$

which concludes the proof. □

Proposition C.2 (CPE as Upper Bound on KL Divergence)**.** *Given a sampling schedule $\{T \to \cdots \to t_{K-1} \to 1\}$, the KL divergence between the sampled distribution $\mathbb{P}_0^v$ and true*

distribution $\mathbb{P}_1^\star$ is upper bounded by the total CPE:

$$D_{KL}(\mathbb{P}_1^\star \| \mathbb{P}_1^v) \leq \sum_{k=0}^{K-1} \left[\mathcal{E}_{CPE}^{ins}(t_k \to t_{k+1}) + \mathcal{E}_{CPE}^{unmsk}(t_k \to t_{k+1}) \right] \tag{70}$$

Proof. First, we recall our definition for the CPE for insertion as:

$$\mathcal{E}_{\text{CPE}}(s \to t) := \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s} \left[D_{\text{KL}} \left(p(\mathcal{I}_t, \boldsymbol{X}_t | \boldsymbol{X}_s) \middle\| p(\mathcal{I}_t | \boldsymbol{X}_s) p(\boldsymbol{X}_t | \boldsymbol{X}_s) \right) \right] \tag{71}$$

and the CPE for unmasking as:

$$\mathcal{E}_{\text{CPE}}^{\text{unmsk}}(s \to t) = \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s} \left[D_{\text{KL}} \left(p(\boldsymbol{X}_t^{\ell_1}, \ldots, \boldsymbol{X}_t^{\ell_n} | \boldsymbol{X}_s) \middle\| \prod_{i=1}^{n} p(\boldsymbol{X}_t^{\ell_i} | \boldsymbol{X}_s) \right) \right] \tag{72}$$

Denoting the joint distribution under the true path measure $\mathbb{P}^\star$ as $\mathbb{P}^\star_{t_{k+1}|t_k}(\cdot|\boldsymbol{x}_{t_k}) := p(\boldsymbol{X}^{\ell_1}_{t_{k+1}}, \ldots, \boldsymbol{X}^{\ell_n}_{t_{k+1}} | \boldsymbol{X}_{t_k})$ and the parameterized marginal distribution as $\mathbb{P}^v_{t_{k+1}|t_k}(\cdot|\boldsymbol{x}_{t_k}) := \prod_{i=1}^n p(\boldsymbol{X}^{\ell_i}_{t_{k+1}} | \boldsymbol{X}_{t_k})$, we can write the terminal distribution generated by both path measures as:

$$\mathbb{P}_0^\star = \mathbb{P}^\star_{t_K|t_{K-1}} \ldots \mathbb{P}^\star_{t_1|t_0} \mathbb{P}^\star_{t_0}, \quad \mathbb{P}_0^v = \mathbb{P}^v_{t_K|t_{K-1}} \ldots \mathbb{P}^v_{t_1|t_0} \mathbb{P}^v_{t_0} \tag{73}$$

Now, using Lemma C.1, we can write the KL divergence between the marginals at time t where $s > t$ as:

$$\begin{aligned}
D_{\text{KL}}(\mathbb{P}_t^\star \| \mathbb{P}_t^v) &\leq D_{\text{KL}}(\mathbb{P}^\star_{s,t} \| \mathbb{P}^v_{s,t}) \\
&= D_{\text{KL}}(\mathbb{P}_s^\star \| \mathbb{P}_s^v) + \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s} \left[D_{\text{KL}}(\mathbb{P}_{t|s}(\cdot|\boldsymbol{x}_s) \| \mathbb{P}'_{t|s}(\cdot|\boldsymbol{x}_s)) \right] \\
&= D_{\text{KL}}(\mathbb{P}_s^\star \| \mathbb{P}_s^v) + \mathbb{E}_{\boldsymbol{x}_s \sim \mathbb{P}_s} \left[\mathcal{E}_{\text{CPE}}^{\text{insert}}(s \to t | \boldsymbol{x}_s) + \mathcal{E}_{\text{CPE}}^{\text{unmask}}(s \to t | \boldsymbol{x}_s) \right] \\
&= D_{\text{KL}}(\mathbb{P}_s^\star \| \mathbb{P}_s^v) + \mathcal{E}_{\text{CPE}}^{\text{insert}}(s \to t) + \mathcal{E}_{\text{CPE}}^{\text{unmask}}(s \to t)
\end{aligned} \tag{74}$$

where the equality holds if and only if $D_{\text{KL}}(\mathbb{P}^\star_{t|s} \| \mathbb{P}^v_{t|s}) = 0$. Then, applying this inequality over all K time steps from $t_0 \to \cdots \to t_{K-1} \to t_K = 1$, we have:

$$\begin{aligned}
D_{\text{KL}}(\mathbb{P}_0^\star \| \mathbb{P}_0^v) &= D_{\text{KL}}(\mathbb{P}^\star_{t_K} \| \mathbb{P}^v_{t_K}) \\
&\leq D_{\text{KL}}(\mathbb{P}^\star_{t_{K-1}} \| \mathbb{P}^v_{t_{K-1}}) + \mathcal{E}_{\text{CPE}}^{\text{insert}}(t_{K-1} \to t_K) + \mathcal{E}_{\text{CPE}}^{\text{unmask}}(t_{K-1} \to t_K) \\
&\vdots \\
&\leq \underbrace{D_{\text{KL}}(\mathbb{P}^\star_{t_0} \| \mathbb{P}^v_{t_0})}_{=0} + \sum_{k=0}^{K-1} \left[\mathcal{E}_{\text{CPE}}^{\text{insert}}(t_k \to t_{k+1}) + \mathcal{E}_{\text{CPE}}^{\text{unmask}}(t_k \to t_{k+1}) \right]
\end{aligned} \tag{75}$$

Since the distribution at time t_0 is the fully empty sequence in any-length MDMs (i.e. $\mathbb{P}^\star_{t_0} = \mathbb{P}^v_{t_0} = \pi_{t_0}$), we have finished the proof. This means that the KL divergence between the generated distribution and the true distribution is zero if and only if the compounding parallelization error over all time steps is zero, i.e. $\mathcal{E}_{\text{CPE}}^{\text{insert}}(t_k \to t_{k+1}) = 0$ and $\mathcal{E}_{\text{CPE}}^{\text{unmask}}(t_k \to t_{k+1}) = 0$ for all $k \in \{0, \ldots, K-1\}$.□

C.7 Radon-Nikodym Derivative Between Joint CTMCs

Proposition C.3 (Radon-Nikodym Derivative of Joint Any-Length CTMCs)**.** *Consider two joint any-length path measures $\mathbb{P}$ and $\mathbb{P}'$ defined by the unmasking rate matrices $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ and the insertion rate matrices $\boldsymbol{R}$ and $\boldsymbol{R}'$. Then, the Radon-Nikodym derivative over the trajectory*

$\boldsymbol{X}_{0:1} = (\boldsymbol{X}_t)_{t\in[0,1]}$ *is defined as:*

$$\begin{aligned}\log \frac{\mathrm{d}\mathbb{P}'}{\mathrm{d}\mathbb{P}}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) \\ + \sum_{t_u:\boldsymbol{X}_{t_u}\neq\boldsymbol{X}_{t_u-}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u-},\boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u-},\boldsymbol{X}_{t_u})} + \int_0^1 \sum_{z\neq\boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z)dt \\ + \sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}} \log \frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i-},\boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i-},\boldsymbol{X}_{t_i})} + \int_0^1 \sum_{y\neq\boldsymbol{X}_t} (\boldsymbol{R}_t - \boldsymbol{R}'_t)(\boldsymbol{X}_t, z)dt \end{aligned} \tag{76}$$

where $t_i \in [0,1]$ denotes the times of insertion events and $t_u \in [0,1]$ denotes the times of unmasking events.

Proof. The discrete time log RND between the CTMC path measures $\mathbb{P}$ and $\mathbb{P}'$ is defined as:

$$\log \frac{\mathrm{d}\mathbb{P}'}{\mathrm{d}\mathbb{P}}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) + \sum_{n=0}^{N-1} \log \frac{\mathrm{d}\mathbb{P}'(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})}{\mathrm{d}\mathbb{P}(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})} + \mathcal{O}(\Delta t) \tag{77}$$

where $\mathbb{P}_0 = \pi_0$ and $\mathbb{P}'_0 = \pi'_0$ are the initial distributions. Now, we have both unmasking and insertion rates given by $\boldsymbol{Q}_t(\boldsymbol{x},\boldsymbol{y})$, which denotes the rate of unmasking from state $\boldsymbol{x} \to \boldsymbol{y}$, and $\boldsymbol{R}_t(\boldsymbol{x},\boldsymbol{y})$, which denotes the rate of insertion from state $\boldsymbol{x} \to \boldsymbol{y}$. Therefore, the total probability of a single jump under the joint path measure $\mathbb{P}$ can be decomposed into the probability of remaining in the same state ($\boldsymbol{y} = \boldsymbol{x}$) and the probability of jumping to a different state ($\boldsymbol{y} \neq \boldsymbol{x}$):

$$\mathbb{P}(\boldsymbol{X}_{t_{n+1}} = \boldsymbol{y}|\boldsymbol{X}_{t_n} = \boldsymbol{x}) = \begin{cases} 1 - \Delta t \sum_{\boldsymbol{z}\neq\boldsymbol{x}} (\boldsymbol{Q}_t(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_t(\boldsymbol{x},\boldsymbol{y})) + \mathcal{O}(\Delta t^2) & \boldsymbol{y} = \boldsymbol{x} \\ \Delta t(\boldsymbol{Q}_t(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_t(\boldsymbol{x},\boldsymbol{y})) + \mathcal{O}(\Delta t^2) & \boldsymbol{y} \neq \boldsymbol{x} \end{cases} \tag{78}$$

For both cases, we will derive the log ratio of the two CTMCs. When the state remains the same over the interval $[t_n, t_{n+1}]$, the log-ratio expands to:

$$\begin{aligned}\log \frac{\mathrm{d}\mathbb{P}'(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})}{\mathrm{d}\mathbb{P}(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})} &= \log \frac{1 - \Delta t \sum_{\boldsymbol{z}\neq\boldsymbol{x}} \left(\boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) + \boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})\right) + \mathcal{O}(\Delta t^2)}{1 - \Delta t \sum_{\boldsymbol{z}\neq\boldsymbol{x}} \left(\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) + \boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})\right) + \mathcal{O}(\Delta t^2)} \\ &= \Delta t \sum_{z\neq\boldsymbol{X}_{t_n}} \big(\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) + \boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) \\ &\quad - \boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) - \boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})\big) + \mathcal{O}(\Delta t^2) \\ &= \Delta t \sum_{z\neq\boldsymbol{X}_{t_n}} \big(\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n},z) - \boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n},z)\big) \\ &\quad + \Delta t \sum_{z\neq\boldsymbol{X}_{t_n}} \big(\boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n},z) - \boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n},z)\big) + \mathcal{O}(\Delta t^2) \end{aligned} \tag{79}$$

When the state changes over $[t_n, t_{n+1}]$, the log-ratio expands to:

$$\begin{aligned}\log \frac{\mathrm{d}\mathbb{P}'(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})}{\mathrm{d}\mathbb{P}(\boldsymbol{X}_{t_{n+1}}|\boldsymbol{X}_{t_n})} &= \log \frac{\Delta t(\boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) + \boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})) + \mathcal{O}(\Delta t^2)}{\Delta t(\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}}) + \boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})) + \mathcal{O}(\Delta t^2)} \\ &= \log \frac{\boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})}{\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})} + \log \frac{\boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})}{\boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n},\boldsymbol{X}_{t_{n+1}})} + \mathcal{O}(\Delta t) \end{aligned} \tag{80}$$

Substituting (79) and (80) into (77) and taking the limit as $\Delta t \to 0$, we get:

$$\begin{aligned}
\log \frac{\mathrm{d}\mathbb{P}'}{\mathrm{d}\mathbb{P}}(\boldsymbol{X}_{0:1}) = \lim_{\Delta t \to 0} \Bigg\{ & \log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) + \sum_{n=0}^{N-1} \log \frac{\boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}})}{\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}})} + \sum_{n=0}^{N-1} \log \frac{\boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}})}{\boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}})} \\
& + \Delta t \sum_{z \neq \boldsymbol{X}_{t_n}} \left(\boldsymbol{Q}_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}}) - \boldsymbol{Q}'_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}}) \right) \\
& + \Delta t \sum_{z \neq \boldsymbol{X}_{t_n}} \left(\boldsymbol{R}_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}}) - \boldsymbol{R}'_{t_n}(\boldsymbol{X}_{t_n}, \boldsymbol{X}_{t_{n+1}}) \right) + \mathcal{O}(\Delta t) \Bigg\} \\
= & \log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) + \sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})} + \int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z) dt \\
& + \sum_{t_i : \boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i -}} \log \frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i -}, \boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i -}, \boldsymbol{X}_{t_i})} + \int_0^1 \sum_{y \neq \boldsymbol{X}_t} (\boldsymbol{R}_t - \boldsymbol{R}'_t)(\boldsymbol{X}_t, z) dt
\end{aligned} \tag{81}$$

which concludes our proof. □

[KL Divergence Between Any-Length Joint CTMCs] The KL divergence between two joint CTMCs $\mathbb{P}, \mathbb{P}'$ with unmasking rates $\boldsymbol{Q}, \boldsymbol{Q}'$ and insertion rates $\boldsymbol{R}, \boldsymbol{R}'$ is given by:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{P}' \| \mathbb{P}) = & D_{\mathrm{KL}}(\mathbb{P}'_0 \| \mathbb{P}_0) \\
& + \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \Bigg[\sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u^-}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u^-}, \boldsymbol{X}_{t_u})} + \int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z) dt \\
& + \sum_{t_i : \boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i -}} \log \frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i^-}, \boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i^-}, \boldsymbol{X}_{t_i})} + \int_0^1 \sum_{y \neq \boldsymbol{X}_t} (\boldsymbol{R}_t - \boldsymbol{R}'_t)(\boldsymbol{X}_t, y) dt \Bigg]
\end{aligned} \tag{82}$$

Proof. The KL divergence is defined as the expectation of the log RND derived in Lemma C.3 as:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{P}' \| \mathbb{P}) &= \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\log \frac{\mathrm{d}\mathbb{P}'}{\mathrm{d}\mathbb{P}}(\boldsymbol{X}_{0:1}) \right] \\
&= \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) \right] \\
&\quad + \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})} + \int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z) dt \right] \\
&\quad + \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\sum_{t_i : \boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i -}} \log \frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i -}, \boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i -}, \boldsymbol{X}_{t_i})} + \int_0^1 \sum_{y \neq \boldsymbol{X}_t} (\boldsymbol{R}_t - \boldsymbol{R}'_t)(\boldsymbol{X}_t, z) dt \right]
\end{aligned} \tag{83}$$

We can decompose the first term as:

$$\mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) \right] = \mathbb{E}_{\boldsymbol{X}_0 \sim \pi_0'} \left[\log \frac{\mathrm{d}\pi_0'}{\mathrm{d}\pi_0}(\boldsymbol{X}_0) \right] = D_{\mathrm{KL}}(\pi_0' \| \pi_0) \tag{84}$$

The second term can be decomposed as:

$$\mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})} + \int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z) dt \right] \tag{85}$$

$$= \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \log \frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u -}, \boldsymbol{X}_{t_u})} \right] + \mathbb{E}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}'} \left[\int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{Q}_t - \boldsymbol{Q}'_t)(\boldsymbol{X}_t, z) dt \right] \tag{86}$$

For the first component, we can consider the time discretization and take the limit for $\Delta t \to 0$ to get:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\left[\sum_{k=0}^{K-1}\mathbf{1}[\boldsymbol{X}_{t_{k+1}}\neq\boldsymbol{X}_{t_k}]\log\frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_k},\boldsymbol{X}_{t_{k+1}})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_k},\boldsymbol{X}_{t_{k+1}})}\right]\\
&\quad=\sum_{k=0}^{K-1}\mathbb{E}_{\mathbb{P}'(\boldsymbol{X}_{t_k}),\mathbb{P}'(\boldsymbol{X}_{t_{k+1}}|\boldsymbol{X}_{t_k})}\left[\mathbf{1}[\boldsymbol{X}_{t_{k+1}}\neq\boldsymbol{X}_{t_k}]\log\frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_k},\boldsymbol{X}_{t_{k+1}})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_k},\boldsymbol{X}_{t_{k+1}})}\right]\\
&\quad=\sum_{k=0}^{K-1}\mathbb{E}_{\mathbb{P}'(\boldsymbol{X}_{t_k})}\sum_{z\neq\boldsymbol{X}_{t_k}}\left[\mathbb{P}'(z|\boldsymbol{X}_{t_k})\log\frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_k},z)}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_k},z)}\right]\\
&\quad=\sum_{k=0}^{K-1}\mathbb{E}_{\mathbb{P}'(\boldsymbol{X}_{t_k})}\sum_{z\neq\boldsymbol{X}_{t_k}}\left[\Delta t\boldsymbol{Q}'_{t_k}(\boldsymbol{X}_{t_k},z)\log\frac{\boldsymbol{Q}'_{t_k}(\boldsymbol{X}_{t_k},z)}{\boldsymbol{Q}_{t_k}(\boldsymbol{X}_{t_k},z)}+\mathcal{O}(\Delta t^2)\right]\\
&\quad\underset{\Delta t\to 0}{=}\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\left[\int_0^1\sum_{z\neq\boldsymbol{X}_t}\boldsymbol{Q}'_t\log\frac{\boldsymbol{Q}'_t}{\boldsymbol{Q}_t}(\boldsymbol{X}_t,z)dt\right]
\end{aligned}\tag{87}$$

Then, summing with the second integral, we get:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\left[\int_0^1\sum_{z\neq\boldsymbol{X}_t}\boldsymbol{Q}'_t\log\frac{\boldsymbol{Q}'_t}{\boldsymbol{Q}_t}(\boldsymbol{X}_t,z)dt\right]+\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\left[\int_0^1\sum_{z\neq\boldsymbol{X}_t}(\boldsymbol{Q}_t-\boldsymbol{Q}'_t)(\boldsymbol{X}_t,z)dt\right]\\
&\quad=\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\int_0^1\sum_{z\neq\boldsymbol{X}_t}\left[\left(\boldsymbol{Q}'_t\log\frac{\boldsymbol{Q}'_t}{\boldsymbol{Q}_t}+\boldsymbol{Q}_t-\boldsymbol{Q}'_t\right)(\boldsymbol{X}_t,z)\right]dt
\end{aligned}\tag{88}$$

Applying the same procedure from lines (85) to (88) to the insertion rates, we get:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\left[\sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}}\log\frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i-},\boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i-},\boldsymbol{X}_{t_i})}+\int_0^1\sum_{y\neq\boldsymbol{X}_t}(\boldsymbol{R}_t-\boldsymbol{R}'_t)(\boldsymbol{X}_t,z)dt\right]\\
&\quad=\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\int_0^1\sum_{z\neq\boldsymbol{X}_t}\left[\left(\boldsymbol{R}'_t\log\frac{\boldsymbol{R}'_t}{\boldsymbol{R}_t}+\boldsymbol{R}_t-\boldsymbol{R}'_t\right)(\boldsymbol{X}_t,z)\right]dt
\end{aligned}\tag{89}$$

Putting these terms together, we get that the KL divergence between the joint CTMCs $\mathbb{P}$ and $\mathbb{P}'$ is given by:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{P}'\|\mathbb{P})&=D_{\mathrm{KL}}(\mathbb{P}'_0\|\mathbb{P}_0)\\
&+\mathbb{E}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}'}\Bigg[\sum_{t_u:\boldsymbol{X}_{t_u}\neq\boldsymbol{X}_{t_u-}}\log\frac{\boldsymbol{Q}'_{t_u}(\boldsymbol{X}_{t_u^-},\boldsymbol{X}_{t_u})}{\boldsymbol{Q}_{t_u}(\boldsymbol{X}_{t_u^-},\boldsymbol{X}_{t_u})}+\int_0^1\sum_{z\neq\boldsymbol{X}_t}(\boldsymbol{Q}_t-\boldsymbol{Q}'_t)(\boldsymbol{X}_t,z)dt\\
&+\sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}}\log\frac{\boldsymbol{R}'_{t_i}(\boldsymbol{X}_{t_i^-},\boldsymbol{X}_{t_i})}{\boldsymbol{R}_{t_i}(\boldsymbol{X}_{t_i^-},\boldsymbol{X}_{t_i})}+\int_0^1\sum_{y\neq\boldsymbol{X}_t}(\boldsymbol{R}_t-\boldsymbol{R}'_t)(\boldsymbol{X}_t,y)dt\Bigg]
\end{aligned}\tag{90}$$

which concludes our proof. □

C.8 Adaptive Joint Decoding Loss

In this section, we provide theoretical justifications for our **Adaptive Joint Decoding (AJD)** loss, which can be used to jointly fine-tune the insertion and unmasking policies, as well as an adaptive inference schedule, to optimize sampling from the reward-tilted distribution. Throughout the proofs, we denote the full any-length generator as $\boldsymbol{A}_t := \boldsymbol{Q}_t + \boldsymbol{R}_t$ where $\boldsymbol{Q}_t$ denotes the unmasking rate and $\boldsymbol{R}_t$ denotes the insertion rate.

We first derive the form of the optimal any-length generator $\boldsymbol{A}^\star$, which follows seamlessly from the theory of fixed-length MDM generators (Zhu et al., 2025; Tang et al., 2025b) but is defined on a larger state space of any-length sequences $\boldsymbol{x} \in \mathcal{X}$. Then, we show that our Adaptive Joint Decoding loss provably converges to the optimal generator.

For an intermediate state $\boldsymbol{x} \in \mathcal{X}$, we define the *total cost* $J_t(\boldsymbol{x}, v)$ of insertions and unmasking steps required to reach a final state $\boldsymbol{X}_1$ under a tilted path measure $\mathbb{P}^v$. Given a terminal reward $r(\boldsymbol{X}_1)$, we define the cost-minimization objective as:

$$J_t(\boldsymbol{x}, v) = \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}^v} \left[\int_0^1 \sum_{y \neq \boldsymbol{X}_s} C_s(\boldsymbol{X}_s, y) ds - r(\boldsymbol{X}_1) \middle| \boldsymbol{X}_t = \boldsymbol{x} \right] \tag{91}$$

where the cost is defined as the KL divergence between generators derived in (C.7) as $C_s(x, y) := \left(\boldsymbol{A}_t^v \log \frac{\boldsymbol{A}^v}{\boldsymbol{A}^0} - \boldsymbol{A}^v + \boldsymbol{A}^0\right)(x, y)$. We can minimize $J_t(\boldsymbol{x}, v)$ by maximizing a **value function** defined as the *negative optimal cost-to-go* $V_t(\boldsymbol{x}) := -J_t^\star(\boldsymbol{x}) = -\inf_v J_t(\boldsymbol{x}, v)$. Using the recursive nature of the cost-to-go, where the total cost is equal to the immediate and future cost, we can decompose the value function using Bellman's principle as:

$$\begin{aligned} -V_t(\boldsymbol{x}) = J_t^\star(\boldsymbol{x}) &= \inf_v \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}^v} \left[\left(\int_t^{t+\Delta t} + \int_{t+\Delta t}^1 \right) \sum_{y \neq \boldsymbol{X}_s} C_s(\boldsymbol{X}_s, y) ds - r(\boldsymbol{X}_1) \middle| \boldsymbol{X}_t = \boldsymbol{x} \right] \\ &= \left[\Delta t \inf_v \sum_{y \neq x} C_s(\boldsymbol{x}, \boldsymbol{y}) + \mathcal{O}(\Delta t^2) \right] + \inf_v \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}^v} [-V_{t+\Delta t}(\boldsymbol{X}_{t+\Delta t}) | \boldsymbol{X}_t = \boldsymbol{x}] \end{aligned} \tag{92}$$

Now, we derive the full form of the optimal any-length generator $\boldsymbol{A}^\star$ using our definition of the value function.

Lemma C.2 (Optimal Generator)**.** *Given a reference any-length generator $\boldsymbol{A}^0 := \boldsymbol{Q}^0 + \boldsymbol{R}^0$ a value function V_t, the optimal generator takes the form:*

$$\begin{aligned} \boldsymbol{A}_t^\star(\boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \exp(V_t(\boldsymbol{y}) - V_t(\boldsymbol{x})) \\ &= \boldsymbol{Q}_t^0(\boldsymbol{x}, \boldsymbol{y}_u) \exp(V_t(\boldsymbol{y}_u) - V_t(\boldsymbol{x})) + \boldsymbol{R}_t^0(\boldsymbol{x}, \boldsymbol{y}^i) \exp(V_t(\boldsymbol{y}^i) - V_t(\boldsymbol{x})) \end{aligned} \tag{93}$$

where $\boldsymbol{y}_u$ denotes the state after an unmasking transition and $\boldsymbol{y}_i$ denotes the state after an insertion transition

Proof. Starting with the expanded value function in (92), we expand the recursive term by defining the next state $\boldsymbol{X}_{t+\Delta t} := \boldsymbol{y}$ and applying the CTMC transition probability as:

$$\begin{aligned} &\inf_v \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1} \sim \mathbb{P}^v} [-V_{t+\Delta t}(\boldsymbol{X}_{t+\Delta t}) | \boldsymbol{X}_t = \boldsymbol{x}] \\ &= \inf_v \left[-\sum_{\boldsymbol{y}} V_{t+\Delta t}(\boldsymbol{y}) \left(\mathbf{1}_{x=y} + \Delta t \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) + \mathcal{O}(\Delta t^2) \right) \right] \\ &= \inf_v \left[-V_{t+\Delta t}(\boldsymbol{x}) - \Delta t \sum_{x \neq y} V_{t+\Delta t}(\boldsymbol{y}) \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) + \Delta t \sum_{x \neq y} V_{t+\Delta t}(\boldsymbol{x}) \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) + \mathcal{O}(\Delta t^2) \right] \\ &= -V_{t+\Delta t}(\boldsymbol{x}) + \Delta t \inf_u \left[\sum_{x \neq y} \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) \left(V_{t+\Delta t}(\boldsymbol{x}) - V_{t+\Delta t}(\boldsymbol{y}) \right) \right] + \mathcal{O}(\Delta t^2)] \end{aligned} \tag{94}$$

Substituting (94) back into the Bellman recursion in (92), we get:

$$\begin{aligned} -V_t(\boldsymbol{x}) = &\left[\Delta t \inf_v \sum_{y \neq x} C_s(\boldsymbol{x}, \boldsymbol{y}) + \mathcal{O}(\Delta t^2) \right] \\ &- V_{t+\Delta t}(\boldsymbol{x}) + \Delta t \inf_u \left[\sum_{x \neq y} \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) \left(V_{t+\Delta t}(\boldsymbol{x}) - V_{t+\Delta t}(\boldsymbol{y}) \right) \right] + \mathcal{O}(\Delta t^2) \\ V_{t+\Delta t}(\boldsymbol{x}) - V_t(\boldsymbol{x}) = &\Delta t \inf_v \sum_{y \neq x} \left[C_s(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) \left(V_{t+\Delta t}(\boldsymbol{x}) - V_{t+\Delta t}(\boldsymbol{y}) \right) \right] \end{aligned} \tag{95}$$

Then, dividing by Δt, taking the limit $\Delta t \to 0$, and substituting $C_s(x, y) = \left(\boldsymbol{A}_t^v \log \frac{\boldsymbol{A}^v}{\boldsymbol{A}^0} - \boldsymbol{A}^v + \boldsymbol{A}^0\right)(x, y)$, we get:

$$\partial_t V_t(\boldsymbol{x}) = \lim_{\Delta t \to 0} \frac{V_{t+\Delta t}(\boldsymbol{x}) - V_t(\boldsymbol{x})}{\Delta t} = \inf_v \sum_{y \neq x} \left[C_s(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) \left(V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})\right)\right] \tag{96}$$

$$= \inf_v \sum_{y \neq x} \left[\left(\boldsymbol{A}_t^v \log \frac{\boldsymbol{A}^v}{\boldsymbol{A}^0} - \boldsymbol{A}^v + \boldsymbol{A}^0\right)(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{A}_t^v(\boldsymbol{x}, \boldsymbol{y}) \left(V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})\right)\right] \tag{97}$$

The infimum is achieved by minimizing the following for every pair $(\boldsymbol{x}, \boldsymbol{y})$:

$$f(\boldsymbol{A}^v) = \boldsymbol{A}_t^v \log \frac{\boldsymbol{A}^v}{\boldsymbol{A}^0} - \boldsymbol{A}^v + \boldsymbol{A}^0 + \boldsymbol{A}_t^v \left(V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})\right) \tag{98}$$

$$f'(\boldsymbol{A}^v) = \log \frac{\boldsymbol{A}^v}{\boldsymbol{A}^0} + \left(V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})\right) \tag{99}$$

The minimizer $\boldsymbol{A}^\star$ satisfying $f'(\boldsymbol{A}^v) = 0$ is defined as:

$$\log \frac{\boldsymbol{A}^\star}{\boldsymbol{A}^0} = V_t(\boldsymbol{y}) - V_t(\boldsymbol{x}) \implies \boldsymbol{A}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \exp\left(V_t(\boldsymbol{y}) - V_t(\boldsymbol{x})\right) \tag{100}$$

Since the total generator factorizes into the sum of insertion and unmasking rates, and rates do not overlap as defined in App C.1, we have:

$$\boldsymbol{A}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{Q}_t^\star(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{R}_t^\star(\boldsymbol{x}, \boldsymbol{y}) \quad \text{s.t.} \quad \begin{cases} \boldsymbol{Q}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{Q}_t^0(\boldsymbol{x}, \boldsymbol{y}) \exp\left(V_t(\boldsymbol{y}) - V_t(\boldsymbol{x})\right) \\ \boldsymbol{R}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{R}_t^0(\boldsymbol{x}, \boldsymbol{y}) \exp\left(V_t(\boldsymbol{y}) - V_t(\boldsymbol{x})\right) \end{cases} \tag{101}$$

which proves our result. □

Now, we will derive the form of the optimal path measure $\mathbb{P}^\star$ corresponding to the optimal generator $\boldsymbol{A}^\star$.

Lemma C.3 (Optimal Path Measure)**.** *The optimal path measure $\mathbb{P}^\star$ corresponding to the value function $V_t(\boldsymbol{x})$ is defined as:*

$$\mathbb{P}_t^\star(\boldsymbol{x}) = \frac{1}{Z} \mathbb{P}_t^0(\boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{x})}, \quad Z := \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_1^0}\left[\mathrm{e}^{r(\boldsymbol{x})/\alpha}\right] \tag{102}$$

for continuous $t \mapsto \boldsymbol{A}_t^0$.

Proof. We define $h_t(\boldsymbol{x}) := \frac{1}{Z} \mathbb{P}_t^0(\boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{x})}$, which satisfies $h_1 = \mathbb{P}_1^\star$ by definition. To show that $\mathbb{P}^\star$ is the path measure generated by $\boldsymbol{A}^\star$, it suffices to show that $h_t(\boldsymbol{x})$ satisfies the Kolmogorov forward equation for $\boldsymbol{A}_t^\star$. First, the Kolmogorov forward equation for the reference generator $\boldsymbol{A}_t^0$ is given by:

$$\partial_t \mathbb{P}_t^0(\boldsymbol{x}) = \sum_{y \neq x} \left(\boldsymbol{A}_t^0(\boldsymbol{y}, \boldsymbol{x}) \mathbb{P}_t^0(\boldsymbol{y}) - \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \mathbb{P}_t^0(\boldsymbol{x})\right) \tag{103}$$

Then, from Lemma C.2, we have $\boldsymbol{A}_t^\star(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \mathrm{e}^{V_t(\boldsymbol{y}) - V_t(\boldsymbol{x})}$ which can be substituted into (97) to get:

$$\partial_t V_t(\boldsymbol{x}) = \sum_{y \neq x} \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \left(\mathrm{e}^{V_t(\boldsymbol{x})} - \mathrm{e}^{V_t(\boldsymbol{y})}\right) \tag{104}$$

Taking the partial derivative of $h_t(\boldsymbol{x})$ and substituting (104), we get:

$$
\begin{aligned}
\partial_t h_t(\boldsymbol{x}) &= \frac{1}{Z}\left[\partial_t \mathbb{P}_t^0(\boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{x})} + \mathbb{P}_t^0 \partial_t \mathrm{e}^{V_t(\boldsymbol{x})}\right] \\
&= \frac{1}{Z}\left[\mathrm{e}^{V_t(\boldsymbol{x})} \sum_{y \neq x} \left(\boldsymbol{A}_t^0(\boldsymbol{y}, \boldsymbol{x}) \mathbb{P}_t^0(y) - \boldsymbol{A}_t^0(\boldsymbol{x}, y) \mathbb{P}_t^0(\boldsymbol{x})\right) + \mathbb{P}_t^0(\boldsymbol{x}) \sum_{y \neq x} \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \left(\mathrm{e}^{V_t(\boldsymbol{x})} - \mathrm{e}^{V_t(\boldsymbol{y})}\right)\right] \\
&= \sum_{y \neq x} \left(\boldsymbol{A}_t^0(\boldsymbol{y}, \boldsymbol{x}) \frac{1}{Z} \mathbb{P}_t^0(\boldsymbol{y}) \mathrm{e}^{V_t(\boldsymbol{x})} - \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \frac{1}{Z} \mathbb{P}_t^0(\boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{y})}\right) \\
&= \sum_{y \neq x} \left(\boldsymbol{A}_t^0(\boldsymbol{y}, \boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})} h_t(\boldsymbol{y}) - \boldsymbol{A}_t^0(\boldsymbol{x}, \boldsymbol{y}) \mathrm{e}^{V_t(\boldsymbol{x}) - V_t(\boldsymbol{y})} h_t(\boldsymbol{x})\right) \\
&\overset{(104)}{=} \sum_{y \neq x} \left(\boldsymbol{A}_t^\star(\boldsymbol{y}, \boldsymbol{x}) h_t(\boldsymbol{y}) - \boldsymbol{A}_t^\star(\boldsymbol{x}, \boldsymbol{y}) h_t(\boldsymbol{x})\right)
\end{aligned}
$$

The final equality is exactly the Kolmogorov forward equation of the optimal any-length generator $\boldsymbol{A}_t^\star$ for the tilted path measure $\mathbb{P}^\star$. Since the Kolmogorov forward equation yields a unique solution for all t, we have shown that $\mathbb{P}_t^\star = \frac{1}{Z} \mathbb{P}_t^0(\boldsymbol{x}) \mathrm{e}^{V_t(\boldsymbol{x})}$. □

Finally, we can derive the Radon-Nikodym derivative between the optimal any-length path measure $\mathbb{P}^\star$ and the base measure $\mathbb{P}^0$ in the following Lemma.

Lemma C.4 (Radon-Nikodym Derivative Between Optimal and Base Path Measures)**.** *The RND between the optimal path measure $\mathbb{P}^\star$ and its generator $\boldsymbol{Q}^\star$ and the reference path measure $\mathbb{P}^0$ and generator $\boldsymbol{Q}^0$ over any trajectory $\boldsymbol{X}_{0:1}$ can be expressed as:*

$$
\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) = \frac{1}{Z} e^{r(\boldsymbol{X}_1)/\alpha}, \quad \textit{where} \quad Z := \mathbb{E}_{\boldsymbol{X}_1 \sim \mathbb{P}_1^0}[e^{r(\boldsymbol{X}_1)/\alpha}] \tag{105}
$$

Proof. From Prop C.3, the RND between the path measures is defined as:

$$
\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\mathbb{P}_0^\star}{\mathrm{d}\mathbb{P}_0^0}(\boldsymbol{X}_0) + \sum_{t: \boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} \log \frac{\boldsymbol{A}_t^\star(\boldsymbol{X}_{t_u-}, \boldsymbol{X}_t)}{\boldsymbol{A}_t^0(\boldsymbol{X}_{t^-}, \boldsymbol{X}_t)} + \int_0^1 \sum_{z \neq \boldsymbol{X}_t} (\boldsymbol{A}_t^0 - \boldsymbol{A}_t^\star)(\boldsymbol{X}_t, z) dt
$$

Then, applying Lemmas C.2 and C.3, we get:

$$
\begin{aligned}
\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) =& V_0(\boldsymbol{X}_0) - \log Z + \sum_{t: \boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} (V_t(\boldsymbol{X}_t) - V_t(\boldsymbol{X}_{t^-})) \\
&+ \int_0^1 \sum_{y \neq \boldsymbol{X}_t} \boldsymbol{A}_t^0(\boldsymbol{X}_t, y) \left(1 - \mathrm{e}^{V_t(\boldsymbol{y}) - V_t(\boldsymbol{X}_t)}\right) dt
\end{aligned} \tag{106}
$$

Now, we derive an expression for $V_0(\boldsymbol{X}_0)$ with respect to $V_1(\boldsymbol{X}_1) = r(\boldsymbol{X}_1)$ by recognizing that the CTMC is a piecewise càdlàg function where each discrete step $t_k \to t_{k+1}$ can be categorized as a time evolution at a fixed state $\boldsymbol{X}_{t_k}$ or a jump from state $\boldsymbol{X}_{t_k}$ to $\boldsymbol{X}_{t_k^-} \neq \boldsymbol{X}_{t_k}$. Given this, we

decompose the value difference as:

$$
\begin{aligned}
V_1(\boldsymbol{X}_1) - V_0(\boldsymbol{X}_0) &= \sum_{k=0}^{K-1} (V_{t_{k+1}}(\boldsymbol{X}_{t_k}) - V_{t_k}(\boldsymbol{X}_{t_k})) + \sum_{k=1}^{K-1} (V_{t_k}(\boldsymbol{X}_{t_k}) - V_{t_k}(\boldsymbol{X}_{t_{k-1}})) \\
&= \sum_{k=0}^{K-1} \int_{t_k}^{t_{k+1}} \partial_t V_t(\boldsymbol{X}_{t_k}) dt + \sum_{t:\boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} (V_t(\boldsymbol{X}_t) - V_t(\boldsymbol{X}_{t^-})) \\
&= \int_0^1 \partial_t V_t(\boldsymbol{X}_t) dt + \sum_{t:\boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} (V_t(\boldsymbol{X}_t) - V_t(\boldsymbol{X}_{t^-})) \qquad (107) \\
V_0(\boldsymbol{X}_0) &= V_1(\boldsymbol{X}_1) - \int_0^1 \partial_t V_t(\boldsymbol{X}_t) dt - \sum_{t:\boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} (V_t(\boldsymbol{X}_t) - V_t(\boldsymbol{X}_{t^-})) \\
&\overset{(C.2)}{=} V_1(\boldsymbol{X}_1) - \int_0^1 \sum_{\boldsymbol{y} \neq \boldsymbol{X}_t} \boldsymbol{A}_t^0(\boldsymbol{X}_t, \boldsymbol{y}) \left(1 - \mathrm{e}^{V_t(\boldsymbol{y}) - V_t(\boldsymbol{X}_t)}\right) dt - \sum_{t:\boldsymbol{X}_t \neq \boldsymbol{X}_{t-}} (V_t(\boldsymbol{X}_t) - V_t(\boldsymbol{X}_{t^-})) \qquad (108)
\end{aligned}
$$

where the final equality follows from Lemma C.2. Substituting this expression for $V_0(\boldsymbol{X}_0)$ into (106), terms cancel and the RND reduces to:

$$
\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) = V_1(\boldsymbol{X}_1) - \log Z \implies \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) = \frac{1}{Z} \mathrm{e}^{V_1(\boldsymbol{X}_1)} \qquad (109)
$$

where $V_1(\boldsymbol{X}_1) = r(\boldsymbol{X}_1)/\alpha$, we conclude our proof. □

Finally, we are ready to derive the RND between the optimal path measure and the parameterized path measure.

Proposition 4.1 (Radon-Nikodym Derivative of Parameterized Rates)**.** *Let the fine-tuned unmasking rate be $f^v(\boldsymbol{x}_t, t)[\ell] \in \Delta^{V-1}$ and the insertion rate be $g^v(\boldsymbol{x}_t, t)[\ell] \in \mathbb{R}_{\geq 0}$ that generates the path measure $\mathbb{P}^v$. Then, given optimal rates $f^{pre}(\boldsymbol{x}_t, t)$ and $g^{pre}(\boldsymbol{x}_t, t)$ and reward function $r : \mathcal{X} \to \mathbb{R}$, the log RND between the optimal joint CTMC and the fine-tuned CTMC over the trajectory $\boldsymbol{X}_{0:1} = (\boldsymbol{X}_t)_{t\in[0,1]}$ is defined as:*

$$
\begin{aligned}
\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) &= \frac{r(\boldsymbol{X}_1)}{\alpha} - \log Z + \sum_{t_u:\boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u-}} \sum_{\ell:\boldsymbol{X}^\ell_{t_u} \neq \boldsymbol{X}^\ell_{t_u-}} \log \frac{f^{pre}(\boldsymbol{X}_{t_u}, t_u)[\ell, \boldsymbol{v}]}{f^v(\boldsymbol{X}_{t_u}, t_u)[\ell, \boldsymbol{v}]} \\
&+ \sum_{t_i:\boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i-}} \sum_{\ell:\boldsymbol{X}^\ell_{t_i} \neq \boldsymbol{X}^\ell_{t_i-}} \log \frac{g^{pre}(\boldsymbol{X}_{t_i}, t_i)[\ell]}{g^v(\boldsymbol{X}_{t_i}, t_i)[\ell]} + \int_0^1 \frac{\dot{\alpha}_t}{1 - \alpha_t} \left(\sum_\ell (g^v - g^{pre})(\boldsymbol{X}_t, t)[\ell] \right) dt \qquad (19)
\end{aligned}
$$

where $t_i \in [0, 1]$ denotes the times of insertion events and $t_u \in [0, 1]$ denotes the times of unmasking events, with t_i^- and t_u^- being the left limits denoting the pre-jump state in the CTMC.

Proof. First, we decompose the RND as:

$$
\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0} \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) + \log \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) \qquad (110)
$$

First, we derive the $\log \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1})$ where $\mathbb{P}^0$ is the path measure of the pretrained model $(f^{\text{pre}}, g^{\text{pre}})$ and $\mathbb{P}^v$ is the path measure generated from the tilted models (f^v, g^v). Let t_i denote the times of insertion events and t_u denote the times of unmasking events. Then, the log RND decomposes into:

$$
\begin{aligned}
\log \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) &= \log \frac{\mathrm{d}\mathbb{P}^0_0}{\mathrm{d}\mathbb{P}^v_0}(\boldsymbol{X}_0) + \sum_{t_u:\boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u-}} \log \frac{\boldsymbol{Q}^0_{t_u}(\boldsymbol{X}_{t_u-}, \boldsymbol{X}_{t_u})}{\boldsymbol{Q}^v_{t_u}(\boldsymbol{X}_{t_u-}, \boldsymbol{X}_{t_u})} + \int_0^1 \sum_{\boldsymbol{z} \neq \boldsymbol{X}_t} (\boldsymbol{Q}^v_t - \boldsymbol{Q}^0_t)(\boldsymbol{X}_t, \boldsymbol{z}) dt \\
&+ \sum_{t_i:\boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i-}} \log \frac{\boldsymbol{R}^0_{t_i}(\boldsymbol{X}_{t_i-}, \boldsymbol{X}_{t_i})}{\boldsymbol{R}^v_{t_i}(\boldsymbol{X}_{t_i-}, \boldsymbol{X}_{t_i})} + \int_0^1 \sum_{\boldsymbol{y} \neq \boldsymbol{X}_t} (\boldsymbol{R}^v_t - \boldsymbol{R}^0_t)(\boldsymbol{X}_t, \boldsymbol{y}) dt \qquad (111)
\end{aligned}
$$

Now, we recall the form of the any-length MDM unmasking rate:

$$\boldsymbol{Q}_t^v(\boldsymbol{x}, \boldsymbol{x}^{\ell \leftarrow \boldsymbol{v}}) = \frac{\dot{\beta}_t}{1-\beta_t} \cdot \mathbb{P}(\boldsymbol{x}_1^{s_t[i]} = \boldsymbol{v} | \boldsymbol{x}_t = \boldsymbol{x}), \quad \boldsymbol{v} \in \mathcal{V}, \boldsymbol{x}^i = \boldsymbol{M} \tag{112}$$

$$= \frac{\dot{\beta}_t}{1-\beta_t} \cdot f^v(\boldsymbol{x}, t)[i, \boldsymbol{v}] \tag{113}$$

where the sum over the vocabulary $\sum_{d \in \mathcal{V}} f_\theta(\boldsymbol{x}, t)[i, \boldsymbol{v}] = 1$. For a state $\boldsymbol{x}$ with $|\{\ell : \boldsymbol{x}^\ell = \boldsymbol{M}\}|$ masked positions, we have that the **exit rate** from state $\boldsymbol{x}$ reduces to:

$$\sum_{\boldsymbol{y} \neq \boldsymbol{x}} \boldsymbol{Q}_t^v(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\ell : \boldsymbol{x}^\ell = \boldsymbol{M}} \sum_d \boldsymbol{Q}_t^v(\boldsymbol{x}, \boldsymbol{x}^{\ell \leftarrow \boldsymbol{v}}) = \frac{\dot{\beta}_t}{1-\beta_t} \sum_{\ell : \boldsymbol{x}^\ell = \boldsymbol{M}} 1 = \frac{\dot{\beta}_t}{1-\beta_t} \big| \{\ell : \boldsymbol{x}^\ell = \boldsymbol{M}\} \big| \tag{114}$$

Given that the noise schedule β_t is equal for both the pretrained and fine-tuned models, the number of masked positions along the interpolant is equal. So we can write:

$$\sum_{\boldsymbol{y} \neq \boldsymbol{x}} \boldsymbol{Q}_t^0(\boldsymbol{x}, \boldsymbol{y}) = \frac{\dot{\beta}_t}{1-\beta_t} \big| \{\ell : \boldsymbol{x}^\ell = \boldsymbol{M}\} \big| \tag{115}$$

and the exit term vanishes:

$$\int_0^1 \sum_{y \neq \boldsymbol{X}_{t_u}} (\boldsymbol{Q}_{t_u}^v - \boldsymbol{Q}_{t_u}^0)(\boldsymbol{X}_{t_u}, y) \mathrm{d}t = 0 \tag{116}$$

The insertion rate is defined as:

$$\boldsymbol{R}_t^v(\boldsymbol{x}, \boldsymbol{x}^{\triangleleft_\ell \boldsymbol{M}}) = \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot \mathbb{E}[s_t[\ell] - s_t[\ell-1] - 1 | \boldsymbol{x}_t = \boldsymbol{x}] \tag{117}$$

$$= \frac{\dot{\alpha}_t}{1-\alpha_t} g^v(\boldsymbol{x}, t)[\ell] \tag{118}$$

Since the insertion rate depends on the model outputs, we have that $\boldsymbol{R}_t^0(\boldsymbol{x}, \boldsymbol{x}^{\triangleleft_\ell \boldsymbol{M}}) \neq \boldsymbol{R}_t^v(\boldsymbol{x}, \boldsymbol{x}^{\triangleleft_\ell \boldsymbol{M}})$. In this case, the exit term is defined as:

$$\int_0^1 \sum_{y \neq \boldsymbol{X}_{t_u}} (\boldsymbol{R}_t^v - \boldsymbol{R}_t^0)(\boldsymbol{X}_t, y) dt = \int_0^1 \frac{\dot{\alpha}_t}{1-\alpha_t} \left(\sum_\ell g^v(\boldsymbol{X}_t, t)[\ell] - \sum_\ell g_\theta^{\text{pre}}(\boldsymbol{X}_t, t)[\ell] \right) dt \tag{119}$$

Putting these terms together, we get the final form of the log RND between any-length path measures:

$$\log \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) = \sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \sum_{\ell : \boldsymbol{X}_{t_u -}^\ell \neq \boldsymbol{X}_{t_u}^\ell} \log \frac{f^{\text{pre}}(\boldsymbol{X}_{t_u}, t)[\ell, \boldsymbol{v}]}{f^v(\boldsymbol{X}_{t_u}, t)[\ell, \boldsymbol{v}]}$$

$$+ \sum_{t_i : \boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i -}} \sum_{\ell : \boldsymbol{X}_{t_i} = \boldsymbol{X}_{t_i^-}^{\triangleleft_\ell \boldsymbol{M}}} \log \frac{g_\theta^{\text{pre}}(\boldsymbol{X}_{t_i}, t_i)[\ell]}{g^v(\boldsymbol{X}_{t_i}, t_i)[\ell]} + \int_0^1 \frac{\dot{\alpha}_t}{1-\alpha_t} \left(\sum_\ell g^v(\boldsymbol{X}_t, t)[\ell] - \sum_\ell g^{\text{pre}}(\boldsymbol{X}_t, t)[\ell] \right) dt \tag{120}$$

Finally, substituting the result from Lemma C.4 and (120) into (110), we get:

$$\log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1}) = \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^0}(\boldsymbol{X}_{0:1}) + \log \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1})$$

$$= \frac{r(\boldsymbol{X}_1)}{\alpha} - \log Z + \sum_{t_u : \boldsymbol{X}_{t_u} \neq \boldsymbol{X}_{t_u -}} \sum_{\ell : \boldsymbol{X}_{t_u}^\ell \neq \boldsymbol{X}_{t_u^-}^\ell} \log \frac{f^{\text{pre}}(\boldsymbol{X}_{t_u}, t)[\ell, \boldsymbol{v}]}{f^v(\boldsymbol{X}_{t_u}, t)[\ell, \boldsymbol{v}]}$$

$$+ \sum_{t_i : \boldsymbol{X}_{t_i} \neq \boldsymbol{X}_{t_i -}} \sum_{\ell : \boldsymbol{X}_{t_i} = \boldsymbol{X}_{t_i^-}^{\triangleleft_\ell \boldsymbol{M}}} \log \frac{g_\theta^{\text{pre}}(\boldsymbol{X}_{t_i}, t_i)[\ell]}{g^v(\boldsymbol{X}_{t_i}, t_i)[\ell]}$$

$$+ \int_0^1 \frac{\dot{\alpha}_t}{1-\alpha_t} \left(\sum_\ell g^v(\boldsymbol{X}_t, t)[\ell] - \sum_\ell g^{\text{pre}}(\boldsymbol{X}_t, t)[\ell] \right) dt \tag{121}$$

which concludes our proof. □

We use this form of the log RND to define the importance weight $W^v(\boldsymbol{X}_{0:1}) := \log \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1})$ in the AJD loss, which yields the optimal any-length path measure $\mathbb{P}^\star$, as we will prove in Prop C.4 below.

Proposition C.4 (Adaptive Joint Decoding Loss)**.** *The unique minimizer of the* ***Adaptive Joint Decoding (AJD)*** *loss defined as:*

$$\mathcal{L}_{AJD}(\theta,\phi) := \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}^v}\left[\frac{1}{Z}\mathrm{e}^{W^v}\left[\mathcal{L}_{unmask}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{insert}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{UQL}(\phi;\boldsymbol{X}_1)+\mathcal{L}_{IQL}(\phi;\boldsymbol{X}_1)\right]\right]$$

is the optimal unmasking generator $\boldsymbol{Q}^\star$ and insertion generator $\boldsymbol{R}^\star$ of the reward-tilted path measure $\mathbb{P}^\star$

Proof. First, we recall the form of the cross-entropy loss as:

$$\mathcal{F}_{\mathrm{CE}}(\mathbb{P}^{\theta,\phi},\mathbb{P}^\star) := D_{\mathrm{KL}}(\mathbb{P}^\star\|\mathbb{P}^{\theta,\phi}) = \mathbb{E}_{\mathbb{P}^\star}\left[\log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] = \mathbb{E}_{\mathbb{P}^v}\left[\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}\log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] \tag{122}$$

Then, we substitute $W^v(\boldsymbol{X}_{0:1}) = \log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}(\boldsymbol{X}_{0:1})$. Since $\mathbb{P}^\star$ is fixed under (θ,ϕ), minimizing the CE loss is equivlalent to minimizing:

$$\min_{\theta,\phi}\mathbb{E}_{\mathbb{P}^v}\left[\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}\log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right] = \min_{\theta,\phi}\mathbb{E}_{\mathbb{P}^v}\left[\frac{1}{Z}\mathrm{e}^{W^v(\boldsymbol{X}_{0:1})}(-\log\mathrm{d}\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1}))\right] \tag{123}$$

Since the path measure $\mathbb{P}^{\theta,\phi}$ is defined by disjoint insertion and unmasking decisions, the probability of a path $\boldsymbol{X}_{0:1}$ under $\mathbb{P}^{\theta,\phi}$ can be written as the product of insertion and unmasking rates:

$$\begin{aligned}\mathrm{d}\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1}) &= \mathbb{P}_0^{\theta,\phi}(\boldsymbol{X}_0)\prod_{t:\boldsymbol{X}_{t^-}\neq\boldsymbol{X}_t}\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t^-}|\boldsymbol{X}_t)\\ &= \mathbb{P}_0^{\theta,\phi}(\boldsymbol{X}_0)\prod_{t_i:\boldsymbol{X}_{t_i^-}\neq\boldsymbol{X}_{t_i}}\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_i^-}|\boldsymbol{X}_{t_i})\prod_{t_u:\boldsymbol{X}_{t_u^-}\neq\boldsymbol{X}_{t_u}}\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_u^-}|\boldsymbol{X}_{t_u})\end{aligned} \tag{124}$$

Taking logs, we have:

$$-\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1}) = -\log\mathbb{P}_0^{\theta,\phi}(\boldsymbol{X}_0) + \sum_{t_i:\boldsymbol{X}_{t_i}\neq\boldsymbol{X}_{t_i-}} -\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_i}|\boldsymbol{X}_{t_i^-}) + \sum_{t_u:\boldsymbol{X}_{t_u}\neq\boldsymbol{X}_{t_u-}} -\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{t_u}|\boldsymbol{X}_{t_u^-})$$

Instead of defining the loss with the probability of generating $\boldsymbol{X}_1 \sim p_{\text{target}}$ via only the single trajectory $\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1})$, we can define it with the expectation over many possible trajectories by taking an expectation over intermediate samples from the endpoint-conditioned interpolant $\tilde{\boldsymbol{x}}_t \sim p_t(\cdot|\boldsymbol{X}_1)$ and times $t\sim\mathcal{U}(0,1)$ following (Zhu et al., 2025). Then, we have:

$$-\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1}) = -\log\mathbb{P}_0^{\theta,\phi}(\boldsymbol{X}_0) + \mathop{\mathbb{E}}_{t\sim\mathcal{U}(0,1)}\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_t\sim p_t(\cdot|\boldsymbol{X}_1)}\left[-\log\mathbb{P}^{\theta,\phi}_{\text{insert}}(\boldsymbol{X}_{t_i}|\tilde{\boldsymbol{x}}_t,\boldsymbol{X}_1) - \log\mathbb{P}^{\theta,\phi}_{\text{unmask}}(\boldsymbol{X}_{t_u}|\tilde{\boldsymbol{x}}_t,\boldsymbol{X}_1)\right]$$

Since we decompose each step into unmasking via f_θ, remasking via μ_ϕ, insertion via g_θ, deletion via ν_ϕ, and define the losses as a form of the negative log-likelihood of the step given the **ground-truth** sequence $\boldsymbol{X}_1 \sim p_{\text{target}}$, we can write:

$$-\log\mathbb{P}^{\theta,\phi}(\boldsymbol{X}_{0:1}) = \underbrace{\mathcal{L}_{\text{unmask}}(\theta;\boldsymbol{X}_1)}_{(3)} + \underbrace{\mathcal{L}_{\text{insert}}(\theta;\boldsymbol{X}_1)}_{(2)} + \underbrace{\mathcal{L}_{\text{UQL}}(\phi;\boldsymbol{X}_1)}_{(6)} + \underbrace{\mathcal{L}_{\text{IQL}}(\phi;\boldsymbol{X}_1)}_{(13)} + \text{const} \tag{125}$$

Therefore, the minimizer in (123) is equivalent to:

$$\begin{aligned}&\min_{\theta,\phi}\mathbb{E}_{\mathbb{P}^v}\left[\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^v}\log\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^{\theta,\phi}}\right]\\ &\quad= \min_{\theta,\phi}\mathbb{E}_{\mathbb{P}^v}\left[\frac{1}{Z}\mathrm{e}^{W^v(\boldsymbol{X}_{0:1})}\left[\mathcal{L}_{\text{unmask}}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{\text{insert}}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{\text{UQL}}(\phi;\boldsymbol{X}_1)+\mathcal{L}_{\text{IQL}}(\phi;\boldsymbol{X}_1)\right]\right]\end{aligned} \tag{126}$$

and defining $\mathcal{L}_{\text{AJD}}(\theta,\phi)$ as:

$$\mathcal{L}_{\text{AJD}}(\theta,\phi) := \mathop{\mathbb{E}}_{\boldsymbol{X}_{0:1}\sim\mathbb{P}^v}\left[\frac{1}{Z}\mathrm{e}^{W^v}\left[\mathcal{L}_{\text{unmask}}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{\text{insert}}(\theta;\boldsymbol{X}_1)+\mathcal{L}_{\text{UQL}}(\phi;\boldsymbol{X}_1)+\mathcal{L}_{\text{IQL}}(\phi;\boldsymbol{X}_1)\right]\right] \tag{127}$$

concludes our proof. □

Table 4: **Hyperparameters for pre-training any-length peptide and small molecule masked diffusion models.**

Hyperparameter	Small Molecule	Peptides
Hidden dimension	768	768
Attention heads	12	12
Transformer blocks	12	12
Timestep conditioning dimension	128	128
MLP expansion ratio	4	4
Dropout	0.05	0.05
Positional encoding	Rotary (RoPE)	Rotary (RoPE)
Vocabulary size	$1,882$	587
Max sequence length	256	$1,024$
Global batch size	$2,048$	$1,024$
Optimizer	AdamW	AdamW
Gradient accumulation steps	16	4

D Small Molecule Experiment Details

D.1 Pre-Training the Any-Length Small-Molecule MDM

Data We represent molecules using SAFE (Sequential Attachment-based Fragment Embedding) strings using the SAFE tokenizer with a vocabulary size of $V = 1880$, including the special pad ($\texttt{id} = 3$) and mask ($\texttt{id} = 4$) tokens (Noutahi et al., 2024). All sequences are tokenized with padding and truncation to a maximum length of $L = 256$ tokens.

The model is pretrained on the SAFE-GPT dataset (`datamol-io/safe-gpt`) (Noutahi et al., 2024) hosted on HuggingFace, which contains over 1 billion drug-like molecules sourced from ZINC20 and other chemical databases. The dataset is split into $945,455,307$ training sequences, $118,451,032$ validation sequences, and $118,890,444$ test sequences.

Model Architecture The flexible-length model architecture is a 96M-parameter bidirectional transformer based on the Diffusion Transformer (DiT) adapted for discrete masked diffusion over variable-length sequences. The architecture consists of **(1) token embedding layer** that projects the vocabulary with $V = 1880$ to embeddings of dimension $d = 768$, **(2) timestep conditioning** that encodes the scalar timestep $t \in [0, 1]$ via sinusoidal positional embeddings (frequency dimension 256), projected through a 2-layer MLP with SiLU activation to a conditioning vector of dimension $d_c = 128$, and **(3) Transformer backbone** with 12 DiT blocks with Rotary Position Embeddings (RoPE, base frequency 10,000) and FlexAttention with block masking for variable-length sequence support. There are two prediction heads: **(1) token prediction head** that predicts the unmasking posterior in the form of logits over the vocabulary dimension and **(2) length prediction head** which predicts the per-position expected gap count. The unmasking quality predictor μ_ϕ and insertion quality predictor ν_ϕ are lightweight heads that operate on the final-layer hidden states of the shared backbone, where each applies a LayerNorm followed by a 2-layer MLP (dimensions $d \to d \to 1$) with a GELU activation, producing a per-position scalar that is passed through a sigmoid to yield a quality value in $[0, 1]$. Hyperparameters are provided in Table 4.

Training Details The model is trained for $87,000$ gradient steps using AdamW with a learning rate of 3×10^{-4} and weight decay of 0.03. The learning rate follows a linear warmup over the first 2,000 steps from 3×10^{-10} to 3×10^{-4} followed by cosine annealing to zero over the remaining steps. Training uses a global batch size of $2,048$, implemented with a per-GPU batch size of 64 across 2 GPUs with 16 gradient accumulation steps. An exponential moving average (EMA) of model parameters is maintained with decay rate 0.9999.

D.2 Rewards and Evaluation Metrics

Uniqueness The fraction of valid generated molecules whose canonical SMILES strings are distinct. Higher uniqueness indicates that the model has not collapsed onto a small set of high-reward modes.

Table 5: **Hyperparameters for small molecule and peptide fine-tuning.** The same base hyperparameters were used for all A2D2 baselines as well as fixed-length RL fine-tuning results.

Hyperparameter	Small Molecule	Peptides
Number of Replicates R	16	8
Buffer Size B	1000	1000
Buffer Refresh Fraction	0.3	0.3
Iterations Between Buffer Generation N_{resample}	10	10
Gradient Steps Per Iteration N_{update}	25	10
Alternation Frequency N_{alt}	5	5
Warmup Iterations N_{warmup}	20	20
Number of Sampling Steps N_{steps}	256	256
Training Mini Batch Size	20	10
Reward Scaling α	0.01	0.1
Max Length L	256	256

QED The **Q**uantitative **E**stimate of **D**rug likeness (QED) (Bickerton et al., 2012) is a scalar in $[0, 1]$ that aggregates eight physicochemical properties (molecular weight, logP, hydrogen-bond donors and acceptors, polar surface area, rotatable bonds, aromatic rings, and structural alerts) into a single drug-likeness score. We compute QED using the RDKit implementation and use it both as a fine-tuning reward and as an evaluation metric.

Synthetic Accessibility The Synthetic Accessibility (SA) score (Ertl and Schuffenhauer, 2009) estimates the ease of synthesizing a molecule on a scale from 1 (easy) to 10 (hard), based on fragment contributions and structural complexity penalties. Since the reward is maximized and summed with QED, SA is flipped and rescaled so that higher = more synthesizable. The SA reward is transformed as $\text{reward}_{\text{SA}} = \frac{1}{1+\text{SA}}$. Empirically, we found that optimizing only QED without the SA in the reward also led to a decrease in SA. We report SA via RDKit, with lower values preferred.

Quality Following Jin et al. (2020), we report *quality* as the fraction of generated molecules that are simultaneously valid, unique, drug-like, and synthesizable. A molecule is considered drug-like if its QED (Bickerton et al., 2012) satisfies QED ≥ 0.6 and synthesizable if its SA score (Ertl and Schuffenhauer, 2009) satisfies SA ≤ 4. Quality thus collapses validity, uniqueness, drug-likeness, and synthesizability into a single metric that measures whether the model can generate chemically reasonable and distinct molecules.

Diversity The average pairwise Tanimoto distance ($1 - \texttt{Tanimoto similarity}$) between Morgan fingerprints (radius 2 and 2048 bits) of all valid generated molecules. Higher diversity indicates broader coverage of chemical space.

D.3 Fine-Tuning Details

Fine-tuning with A2D2 follows Algorithm 1 with hyperparameters provided in Table 5. For the A2D2 baseline without quality, we only perform the first loop in Algorithm 1, which fine-tunes the policy model without the planner and does not alternate to the second loop. For the baselines with only insertion or unmasking quality, we alternate to the planner fine-tuning but using only quality-based unmasking *or* quality-based insertions. During buffer generation, we retain only the sequences that pass the quality threshold and weight them by their log RND added to the QED and SA rewards. Fine-tuning was performed on 1 NVIDIA A100 GPU per run.

E Peptide Experiment Details

E.1 Pre-Training the Any-Length Peptide MDM

Data The any-length peptide MDM is pretrained on a corpus of approximately 11 million peptide SMILES sequences from Tang et al. (2025a), containing 7451 sequences from the CycPeptMPDB database (Li et al., 2023), 825,632 unique peptides from SmProt (Li et al., 2021), and approximately

10 million modified peptides generated from CycloPs (Duffy et al., 2011; Feller and Wilke, 2025). We leverage the SMILES Pair Encoding tokenizer with vocabulary size $V = 587$ (Feller and Wilke, 2025). The data is split with a $0.8/0.2$ train/validation ratio and batched with dynamic batch sizes to handle variable-length token sequences, with a per-GPU token count set to $16,000$.

Model Architecture and Training Details We leverage the same model architecture and training procedure as in the small-molecule experiment, with specific hyperparameters provided in Table 4. Pre-training was performed on 4 NVIDIA A100 GPUs until convergence. The checkpoint at 46 epochs was used for fine-tuning.

E.2 Rewards and Evaluation Metrics

Validity The fraction of generated SMILES sequences that can be decoded into a sequence of canonical and non-canonical amino acids using the SMILES2PEPTIDE filter (Tang et al., 2025a).

Uniqueness The fraction of distinct peptides over all valid peptide sequences.

Diversity The similarity of the generated peptide structures, measured as one minus the Tanimoto similarity between the pairwise Morgan fingerprints of generated sequences, computed as:

$$\text{Diversity} = 1 - \frac{1}{\binom{N_{\text{generated}}}{2}} \sum_{i,j} \frac{\boldsymbol{f}(\boldsymbol{x}_i) \cdot \boldsymbol{f}(\boldsymbol{x}_j)}{|\boldsymbol{f}(\boldsymbol{x}_i)| + |\boldsymbol{f}(\boldsymbol{x}_j)| - \boldsymbol{f}(\boldsymbol{x}_i) \cdot \boldsymbol{f}(\boldsymbol{x}_j)} \tag{128}$$

Therapeutic Property Oracles We use lightweight property classifiers from PeptiVerse (Zhang et al., 2026), trained on experimental peptide data, to predict binding affinity to an input protein sequence, solubility, non-hemolysis, non-fouling, and cell permeability. For binding affinity, values > 9 indicate strong nM to pM binders, $7 - 9$ indicates nM to μM medium binders, and < 7 indicates weak μM binders. For solubility, non-hemolysis, and non-fouling, the values are between $[0, 1]$, indicating the probability of the positive class. For cell permeability, the model predicts the logarithm of the effective permeability coefficient $\log P_{\text{exp}}$, where values ≥ -6.0 were labeled as high permeability, and values < -6.0 as weak permeability.

E.3 Fine-Tuning Details

Fine-tuning with A2D2 follows Algorithm 1 with hyperparameters provided in Table 5. Similar to the small molecule experiment, we only perform the first loop in Algorithm 1 for the A2D2 baseline without quality, which fine-tunes the policy model without the planner and does not alternate to the second loop. For the baselines with only insertion or unmasking quality, we alternate to the planner fine-tuning but using only quality-based unmasking *or* quality-based insertions. During buffer generation, we retain only the valid sequences that pass the `SMILES2PEPTIDE` filter (Tang et al., 2025a). The multi-objective rewards are scalarized and added to the log RND for each sequence. Fine-tuning was performed on a single NVIDIA A100 GPU per run.

F Language Reasoning Experiment Details

F.1 Pre-Training the Any-Length Language MDM

Following the pre-training procedure from Kim et al. (2026a), we adapt the LLaDA-8B-Base model (Nie et al., 2025) for any-length generation by adding a time-embedding layer with AdaLN (Peebles and Xie, 2023) since the insertion expectation requires time dependence. Four AdaLN parameter sets are shared across intermediate bidirectional Transformer layers, and the insertion expectation is modeled with an additional `softplus` activation head. To adapt the backbone layers for the new any-length unmasking prediction task while maintaining transfer efficiency, we add LoRA adapters (Hu et al., 2022) to each attention layer and the unmasking posterior prediction head. The LoRA parameters are set to rank $r = 128$, scaling factor $\alpha = 128$, and `dropout` $= 0.1$. The training data consists of an equal mixture of OpenWebText (Gokaslan et al., 2019) and Proof-Pile-2 (Azerbayev et al., 2024), totalling 13.97M training sequences. Training ran for approximately 3 days on 8 B200 NVIDIA GPUs for $240,000$ optimizer steps and ≈ 1.10 epochs.

F.2 Reasoning Tasks

We evaluate on two reasoning tasks where any-length, any-order decoding is particularly well-suited: math word-problem solving and code infilling.

Math reasoning (GSM8K). GSM8K (Cobbe et al., 2021) requires generating a multi-step chain-of-thought before a final numeric answer. Because the length of the reasoning trace varies substantially across problems, fixed-length decoding either truncates long derivations or wastes a large mask budget on padding. Any-length generation instead inserts tokens as the reasoning chain grows, letting the response length adapt to the difficulty of each problem. We prompt the model to emit reasoning and the answer in a fixed XML schema, `<reasoning>...</reasoning><answer>\boxed{...}</answer>`, and extract the boxed value as the predicted answer.

Code infilling (HumanEval-infill) We use the single-line infilling split of the HumanEval benchmark (Chen et al., 2021; Bavarian et al., 2022), where a fixed prefix and suffix surround a masked span that the model must complete. The number of tokens needed to fill the gap is unknown a priori, making this a natural setting for insertion-based decoding: the model conditions bidirectionally on both the prefix and suffix and inserts the variable-length completion between them. We delimit the infilling region with `<infill_boundary>` tokens and score the generated span against the canonical solution.

For both tasks, we compare the A2D2-finetuned any-length model against the pretrained any-length model adapted from LLaMA-8B-Base with instruction fine-tuning under matched inference budgets. Generation uses greedy decoding (temperature 0) with a generation length of 1024, and we sweep the number of diffusion sampling steps $N_{\text{steps}} \in \{128, 256, 512, 1024\}$.

F.3 Instruction Fine-Tuning

We instruction fine-tune both the any-length MDM (obtained from the transfer-learning stage above) and the fixed-length LLaDA-8B-Base baseline separately for each task, with no cross-task data mixing. For math, we fine-tune on the GSM8K train split, formatting each example with the reasoning/answer XML schema described above and supervising only the response tokens. For code, we fine-tune on the `educational_instruct` subset of OpenCoder `opc-sft-stage2` (Huang et al., 2025) and reserve the HumanEval single-line infilling tasks for evaluation. Both models are trained with LoRA adapters of rank $r = 128$, scaling factor $\alpha = 128$, and `dropout` $= 0.1$ applied to the attention projections (`q_proj`, `k_proj`, `v_proj`) and the feed-forward output projection, using a learning rate of 1e-4 with a cosine schedule, a maximum sequence length of 1024, and variable-length batching. We select the checkpoint with the best tracked validation correctness. Instruction fine-tuning was performed on 8 B200 NVIDIA GPUs.

F.4 Reinforcement Learning Fine-Tuning with A2D2

We then fine-tune the instruction-tuned any-length MDM with **A2D2** following Algorithm 1. The insertion–unmasking policy is initialized from the IFT checkpoint and fine-tuned with the Adaptive Joint Decoding (AJD) loss, while the unmasking- and insertion-quality predictor heads are trained from scratch using the same architecture as in the small molecule and peptide experiments. Following Zhou et al. (2025), we alternate between updating the policy and the quality predictors every $N_{\text{alt}} = 5$ iterations, and warm up by sampling the replay buffer with the policy alone for $N_{\text{warmup}} = 50$ iterations before enabling quality-based sampling. Each iteration draws 12 generations per prompt via greedy adaptive inference (temperature 0) at a generation length of 1024, stores them in the replay buffer, and performs 256 gradient steps per epoch with a per-device batch size of 6 and gradient accumulation of 2. The policy is optimized at learning rate 1e-6 with gradient-norm clipping of 0.2 and weight decay 0.1, and the AJD loss uses reward scaling $\alpha = 0.01$, regularization coefficient 0.5, and antithetic sampling of the masking schedule. Buffer generation and the inference-time schedule use quality-guided inference using a sliding window of size 64. Sequences are weighted by their log Radon–Nikodym derivative added to the task reward. RL fine-tuning was performed on 2 B200 NVIDIA GPUs per run. Key hyperparameters are summarized in Table 6.

Table 6: **Hyperparameters for language fine-tuning with A2D2.**

Hyperparameter	GSM8K	HumanEval-infill
Number of Generations	12	12
Iterations Between Buffer Generation N_{resample}	12	12
Gradient Steps Per Iteration N_{update}	256	256
Alternation Frequency N_{alt}	5	5
Warmup Iterations N_{warmup}	50	50
Number of Sampling Steps for Buffer Generation N_{steps}	256	256
Per-Device Batch Size	6	6
Gradient Accumulation Steps	2	2
Policy Learning Rate	1e-6	1e-6
Reward Scaling α	0.01	0.01
Max Length L	1024	1024

F.5 Datasets and Rewards

GSM8K GSM8K (Cobbe et al., 2021) is a benchmark of grade-school arithmetic word problems that test multi-step numerical reasoning. The model is trained on the training partition and evaluated on the held-out test partition. Because A2D2 optimizes the policy through reward signals rather than supervised targets, we require a scalar reward that reflects how well a sampled response solves the problem. We adopt the same reward decomposition as d1 (Zhao et al., 2025a), which jointly rewards correct formatting and correct final answers:

(i) *XML Structure Reward*: If the opening and closing tag (`<reasoning>`,`</reasoning>`, `<answer>`,`</answer>`) is correctly placed, the reward gets $+0.125$ and for each token after the `</answer>` tag, the reward gets -0.001.

(ii) *Soft Format Reward*: If the response matchs the pattern `<reasoning>...</reasoning><answer>...</answer>`, the reward gets $+0.5$.

(iii) *Strict Format Reward*: If the response matches the format exactly with correct line breaks, the reward gets $+0.5$.

(iv) *Integer Answer Reward*: If the extracted value in the answer tags is an integer, the reward gets $+0.5$.

(v) *Correctness Reward*: For answers that are exactly correct, the reward gets $+2$.

HumanEval-infill For the code infilling experiment, we fine-tune on the `educational_instruct` subset of OpenCoder `opc-sft-stage2` (Huang et al., 2025) and evaluate on the single-line infilling split of HumanEval (Chen et al., 2021; Bavarian et al., 2022), where each task provides a prefix, a suffix, and a held-out canonical infill. Since the canonical solution is available as a reference, we define a reward that proxies infilling pass@1 by combining executability with similarity to the reference. Let c denote the generated completion, g the ground-truth infill, and $\texttt{edit_sim}(c, g) = 1 - \frac{\text{Lev}(c,g)}{\max(|c|,|g|)}$ the normalized Levenshtein edit similarity. The reward decomposes as:

(i) *Non-Empty Reward*: If the completion is non-empty, the reward gets $+0.25$.

(ii) *Completion Compile Reward*: If the completion fragment compiles as valid Python, the reward gets $+0.5$, gated on $\texttt{edit_sim}(c, g) > 0.3$ and the ground-truth infill compiling standalone. The edit-similarity gate prevents the model from gaming the reward by emitting arbitrary compiling code.

(iii) *Full-Program Compile Reward*: If the full reconstructed program `(prefix + completion + suffix)` compiles, the reward gets $+0.75$, gated on $\texttt{edit_sim}(c, g) > 0.3$.

(iv) *Edit Similarity Reward*: A soft gradient toward the reference of $+1.5 \cdot \texttt{edit_sim}(c, g)$.

The total reward lies in the range $[0, 3.0]$, comparable to the GSM8K reward range.

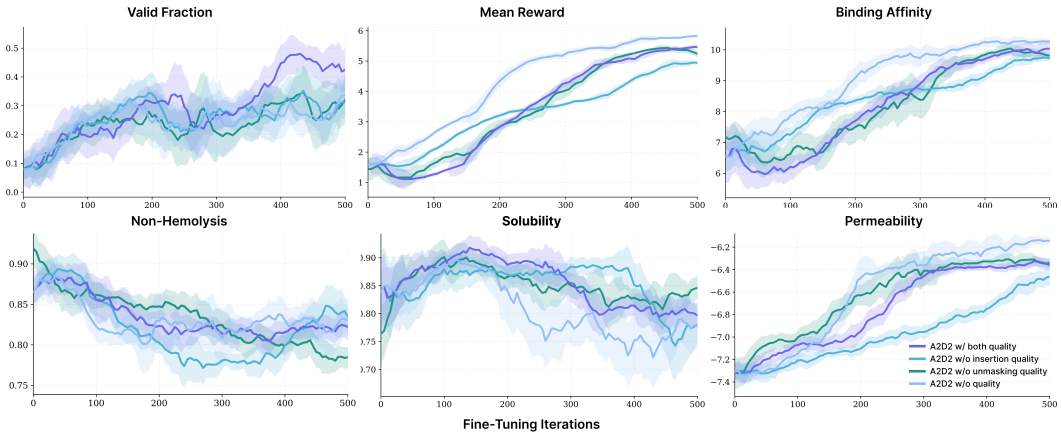

Figure 2: **Multi-objective peptide fine-tuning curves with and without quality-based adaptive inference.** Any-length peptide SMILES models are fine-tuned by optimizing the five rewards: binding affinity to TfR, solubility, non-hemolysis, non-fouling, and permeability. Curves show the fraction of valid peptides, the mean reward, and per-property rewards over fine-tuning iterations for the full model with both insertion and unmasking quality (**indigo**) against ablations that remove the insertion quality (**light blue**), the unmasking quality (**teal**), or both qualities (**pale blue**).

Table 7: **Ablations for *de novo* small molecule design experiment.** Metrics were computed for 1000 generated molecule sequences, and the mean and standard deviations were computed across 3 seeds.

Method	Validity (% total)	Uniqueness (% valid)	Quality (% total)	QED (↑)	Synthetic Accessibility (↓)	Diversity (↑)
Pre-trained (Any Length)	$95.800_{\pm 0.294}$	$99.582_{\pm 0.086}$	$44.167_{\pm 0.574}$	$0.641_{\pm 0.002}$	$3.401_{\pm 0.032}$	$0.908_{\pm 0.000}$
Reward Scaling α						
$\alpha = 0.005$	$97.833_{\pm 0.368}$	$92.982_{\pm 0.363}$	$77.633_{\pm 1.237}$	$0.780_{\pm 0.003}$	$2.670_{\pm 0.009}$	$0.828_{\pm 0.001}$
$\alpha = 0.01$ (default)	$94.533_{\pm 0.665}$	$93.695_{\pm 0.925}$	$71.300_{\pm 0.852}$	$0.762_{\pm 0.004}$	$2.870_{\pm 0.018}$	$0.843_{\pm 0.001}$
Gradient Steps Per Iteration						
10	$97.333_{\pm 0.206}$	$78.938_{\pm 0.299}$	$64.967_{\pm 0.189}$	$0.737_{\pm 0.004}$	$2.733_{\pm 0.023}$	$0.829_{\pm 0.003}$
25 (default)	$94.533_{\pm 0.665}$	$93.695_{\pm 0.925}$	$71.300_{\pm 0.852}$	$0.762_{\pm 0.004}$	$2.870_{\pm 0.018}$	$0.843_{\pm 0.001}$
50	$96.600_{\pm 0.572}$	$88.989_{\pm 0.613}$	$82.633_{\pm 0.618}$	$0.820_{\pm 0.002}$	$2.480_{\pm 0.001}$	$0.771_{\pm 0.002}$

G Ablation Studies

We perform ablations for the small molecule experiment on two key hyperparameters: the KL regularization weight α and the gradient steps per iteration. Results are reported in Table 7. We separately compare training objectives and best-of-N selection in App. H.

KL Regularization Weight The coefficient α controls the trade-off between reward maximization and adherence to the base generative distribution. The smallest value ($\alpha = 0.005$) achieves the highest quality and QED, reflecting strong reward optimization, but at the cost of slightly lower uniqueness and diversity.

Gradient Steps Per Iteration Reducing the number of gradient steps per iteration from 25 to 10 lowers reward values, while validity, uniqueness, and diversity remain comparable, suggesting that the additional steps primarily refine samples toward higher-reward regions rather than affecting coverage of the output space. These results indicate that a sufficient sampling budget is necessary to realize the benefits of reward guidance, and we use 25 steps in all main experiments.

H Additional Baselines

Comparison to Policy-Gradient Objectives We compare policy-training objectives with both quality heads disabled and a matched fine-tuning budget of 36,400 oracle calls. The KL-regularized REINFORCE control uses a leave-one-out baseline and regularization coefficient $\beta = \alpha = 0.01$. Reward-weighted MLE omits the reference-to-proposal likelihood ratio from the weights.

Table 8: **Ablations for multi-objective peptide design.** Metrics were computed for 1000 generated peptide sequences, and the mean and standard deviations were computed across 3 seeds.

Method	Validity (%)	Uniqueness (↑)	Diversity (↑)	Binding Affinity to TfR (↑)	Solubility (↑)	Non-hemolysis (↑)	Non-fouling (↑)	Permeability (↑)
A2D2 w/o quality	$41.267_{\pm 1.520}$	$100.000_{\pm 0.000}$	$0.711_{\pm 0.000}$	$10.241_{\pm 0.012}$	$0.792_{\pm 0.011}$	$0.831_{\pm 0.004}$	$0.096_{\pm 0.001}$	$-6.102_{\pm 0.006}$
A2D2 w/o insertion quality	$39.267_{\pm 1.886}$	$100.000_{\pm 0.000}$	$0.732_{\pm 0.004}$	$9.746_{\pm 0.017}$	$0.772_{\pm 0.007}$	$0.837_{\pm 0.003}$	$0.113_{\pm 0.001}$	$-6.418_{\pm 0.025}$
A2D2 w/o unmasking quality	$31.600_{\pm 1.219}$	$100.000_{\pm 0.000}$	$0.603_{\pm 0.005}$	$9.670_{\pm 0.058}$	$0.864_{\pm 0.009}$	$0.777_{\pm 0.004}$	$0.119_{\pm 0.001}$	$-6.337_{\pm 0.016}$
A2D2 w/ both quality	$48.600_{\pm 1.445}$	$99.868_{\pm 0.187}$	$0.578_{\pm 0.003}$	$10.190_{\pm 0.033}$	$0.776_{\pm 0.003}$	$0.831_{\pm 0.003}$	$0.118_{\pm 0.002}$	$-6.338_{\pm 0.003}$

Table 9: **Training-objective comparisons.** 1000 molecules per reported evaluation.

Objective	Validity (%)	Quality (%)	QED ↑	SA ↓
Pre-trained (Any)	95.8	44.2	0.641	3.401
REINFORCE + KL	94.5	30.5	0.575	3.247
Reward-weighted MLE	88.7	33.3	0.622	3.055
A2D2, no heads	96.4	76.1	0.795	2.306

Best-of-N Selection BoN is competitive on molecules and composes with fine-tuning. Within each model's sweep, diversity decreases as N increases. A2D2 without selection needs no reward-oracle calls at inference, but incurs a training cost. Notably, composing A2D2 with BoN sampling yields the highest quality over all sampling methods.

Table 10: **Molecule best-of-N sweep.** 1000 outputs per row, selecting by QED plus transformed SA. Oracle counts refer to inference-time selection per returned output and exclude training and evaluation.

Model	N	Oracle	Valid (%)	Unique (%)	Quality (%)	QED	SA	Diversity
Pre-trained	1	1	96.0	98.9	43.4	0.643	3.438	0.908
Pre-trained + BoN	2	2	99.7	99.8	61.0	0.716	3.209	0.899
Pre-trained + BoN	4	4	100.0	99.9	78.8	0.777	2.984	0.888
Pre-trained + BoN	8	8	100.0	99.8	89.0	0.822	2.801	0.880
A2D2	1	0	97.7	82.4	64.5	0.779	2.680	0.833
A2D2 + BoN	2	2	99.8	91.4	84.7	0.851	2.654	0.808
A2D2 + BoN	4	4	100.0	93.8	92.9	0.905	2.637	0.788
A2D2 + BoN	8	8	100.0	92.9	92.9	0.926	2.492	0.774

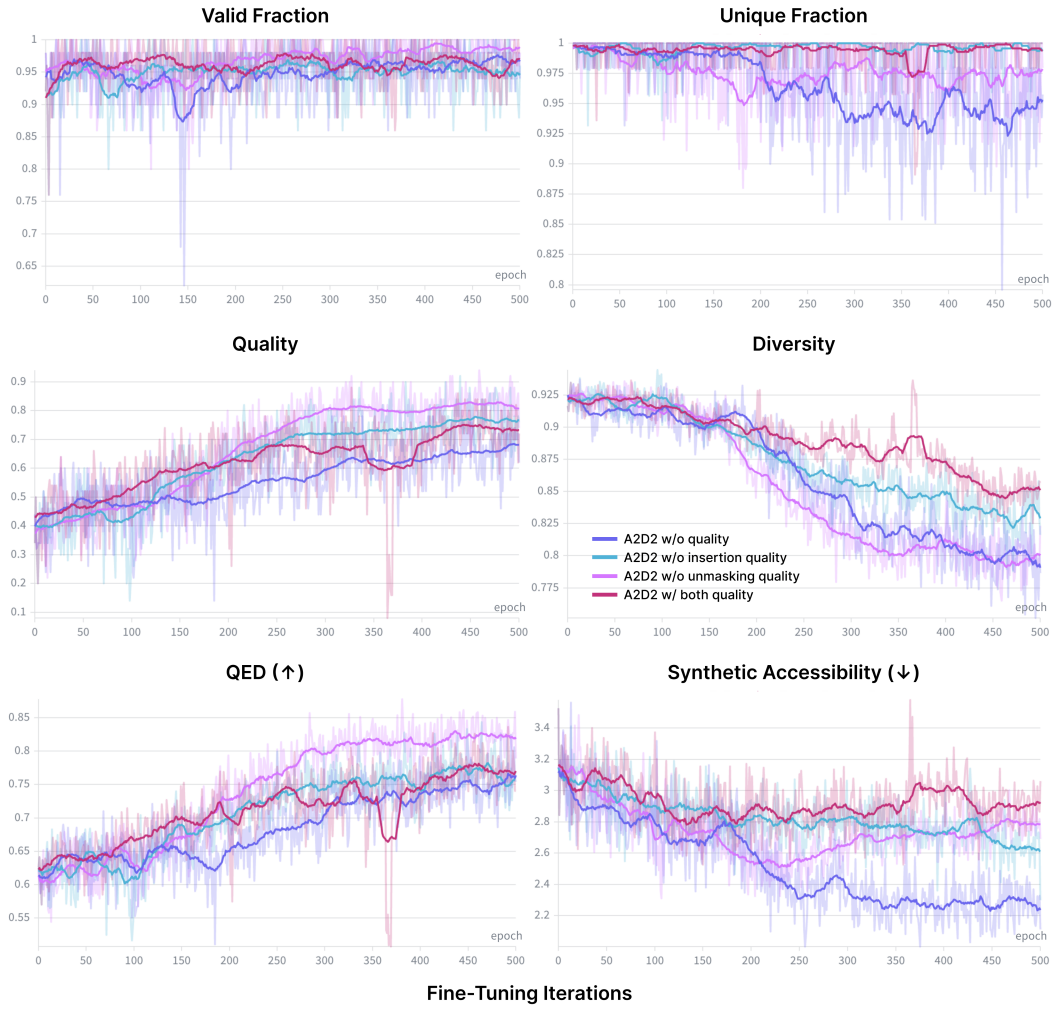

Figure 3: **Ablation study on insertion and unmasking quality for the small molecule experiment.** Average reward values of 50 sequences sampled from the fine-tuned model after each fine-tuning epoch are plotted over a total of 500 iterations, and a running average is shown with the smooth line.

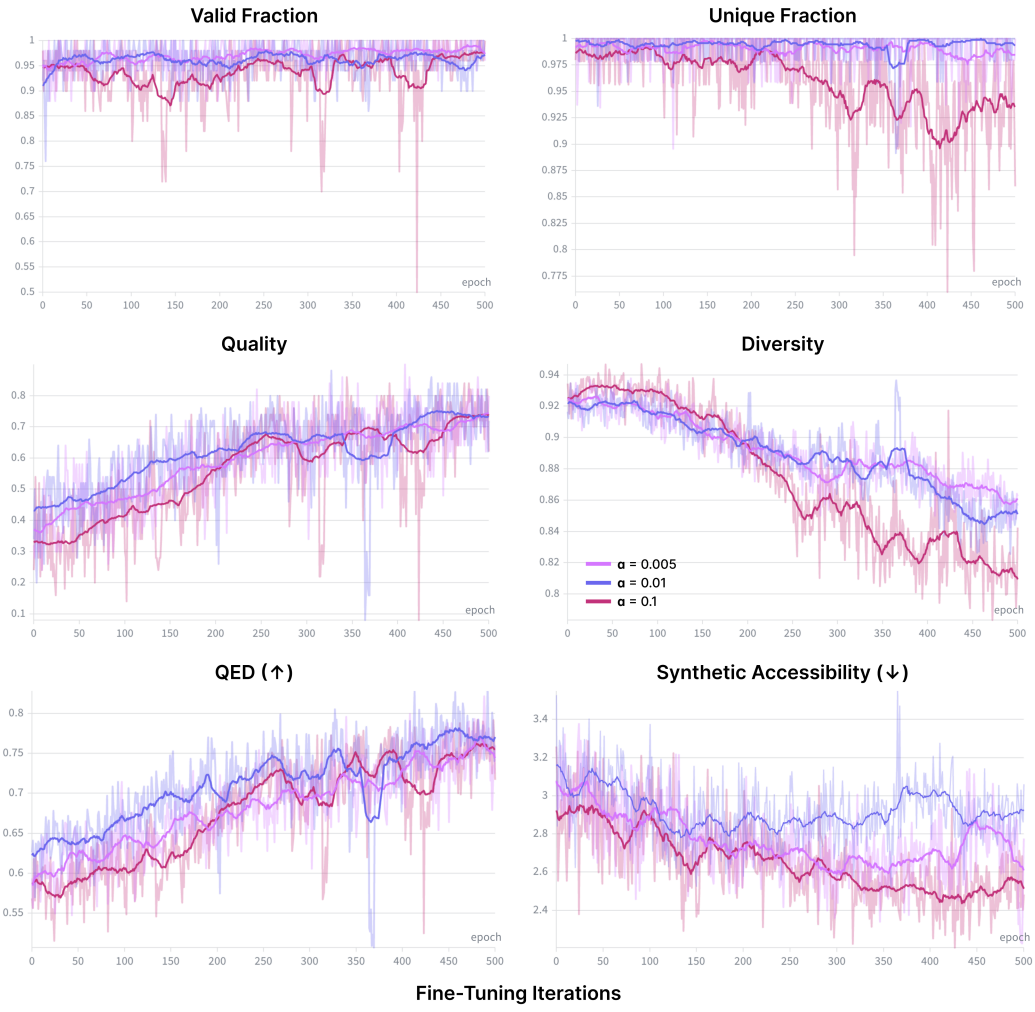

Figure 4: **Ablation study on reward scaling α for the small molecule experiment.** Average reward values of 50 sequences sampled from the fine-tuned model after each fine-tuning epoch are plotted over a total of 500 iterations, and a running average is shown with the smooth line.

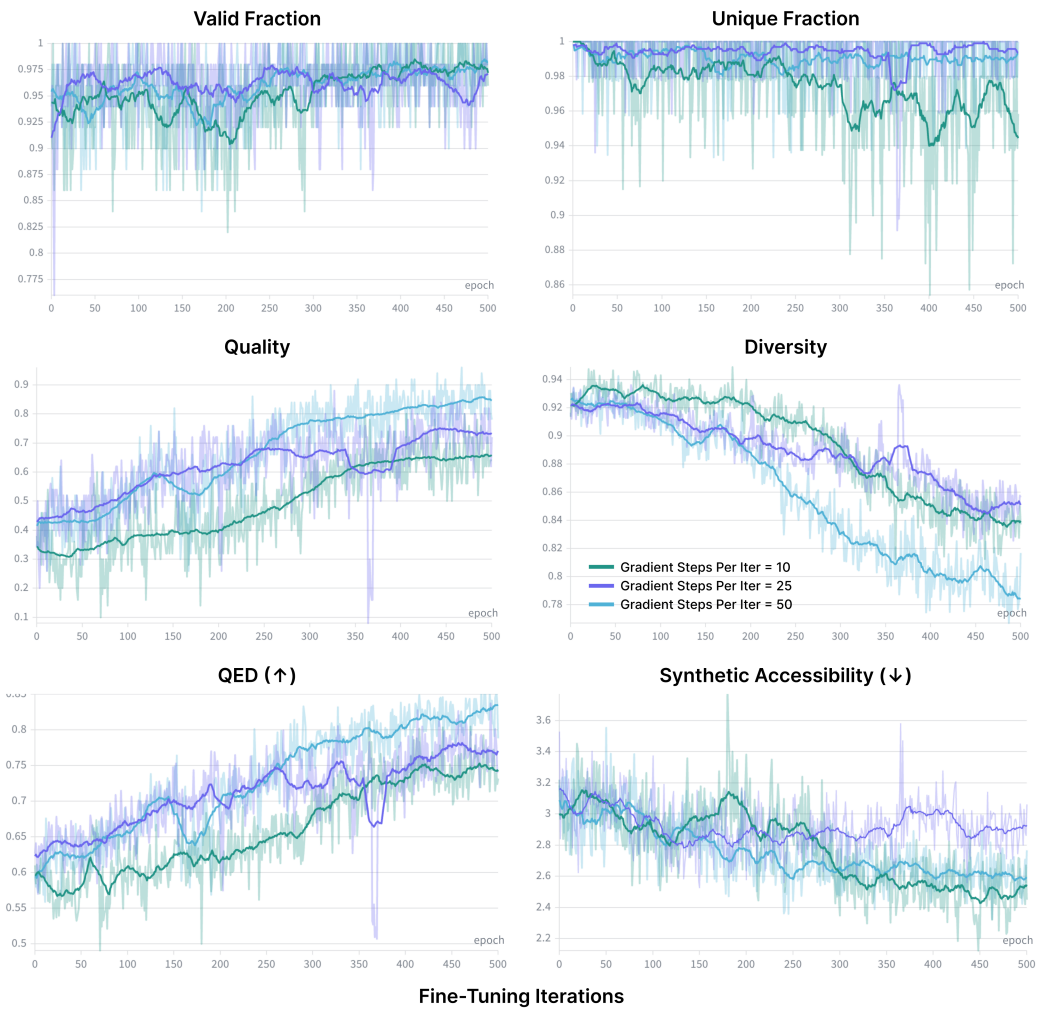

Figure 5: **Ablation study on the number of gradient steps per iteration for the small molecule experiment.** Average reward values of 50 sequences sampled from the fine-tuned model after each fine-tuning epoch are plotted over a total of 500 iterations, and a running average is shown with the smooth line.

I ALGORITHMS

Here, we provide the complete pseudo-code for the algorithms used in the **A2D2** framework. Algorithm 1 describes the joint fine-tuning procedure that alternates between fine-tuning the policy model and the quality planner model to generate from a target reward-tilted distribution. Algorithm 2 describes the adaptive any-length inference procedure using our trained quality predictors. Algorithm 3 describes the method used to sample an intermediate state $\boldsymbol{x}_t$ from the interpolant given a clean sequence $\boldsymbol{x}_1 \sim p_{\text{target}}$. Algorithm 4 describes the procedure for adaptive remasking, and Algorithm 5 describes the procedure for adaptive deletion.

Algorithm 1 `A2D2`: Adaptive Any-Length Discrete Diffusion

Input: Pretrained model $f^{\text{pre}}(\boldsymbol{x}_t, t), g^{\text{pre}}(\boldsymbol{x}_t, t)$, fine-tuned model $f_\theta(\boldsymbol{x}_t, t), g_\theta(\boldsymbol{x}_t, t)$, alternation frequency N_{alt}, buffer size B, number of repeats R
while not converged **do**
  $\{\boldsymbol{x}_i^\star, W^{\bar{\theta},\bar{\phi}}\}_{i=1}^B \leftarrow \texttt{BatchQualitySample}(f^{\text{pre}}, g^{\text{pre}}, f_\theta, g_\theta)$
  $\mathcal{B} \leftarrow \{\boldsymbol{x}_{t,i}, W^{\bar{\theta},\bar{\phi}}\}_{i=1}^B$
  for `epoch` in $1, \ldots, N_{\text{alt}}$ **do**
    ▷ *Fine-tune policy model with frozen planner* ◁
    $\{\tilde{\boldsymbol{x}}_{t,i}, W^{\bar{\theta},\bar{\phi}}\} \leftarrow \texttt{SampleInterpolant}(\mathcal{B}; R)$
    Predict unmasking posterior $f_\theta(\tilde{\boldsymbol{x}}_{t,i}, t)$
    $\mathcal{L}_{\text{unmask}}(\theta; \tilde{\boldsymbol{x}}_{t,i}) \leftarrow -\frac{\dot{\beta}_t}{1-\beta_t} \sum_{\ell:\tilde{\boldsymbol{x}}_{t,j}=\boldsymbol{M}} \log f_\theta(\tilde{\boldsymbol{x}}_{t,i}, t)[\ell, \boldsymbol{x}_i^\star]$
    Predict insertion expectation $g_\theta(\tilde{\boldsymbol{x}}_{t,i}, t)$
    $\mathcal{L}_{\text{insert}}(\theta; \tilde{\boldsymbol{x}}_{t,i}) \leftarrow -\frac{\dot{\alpha}_t}{1-\alpha_t} \sum_{\ell=1}^{\text{len}(\tilde{\boldsymbol{x}}_{t,j})+1} \phi\left(s_t[\ell] - s_t[\ell-1] - 1, g_\theta(\tilde{\boldsymbol{x}}_{t,i}, t)[\ell]\right)$
    $\mathcal{L}_{\text{AJD}}(\theta) \leftarrow \frac{1}{BR} \sum_{(\tilde{\boldsymbol{x}}_{t,i}, W^{\bar{\theta},\bar{\phi}}) \sim \mathcal{B}} \left[e^{W^{\bar{\theta},\bar{\phi}}} (\mathcal{L}_{\text{unmask}}(\theta; \tilde{\boldsymbol{x}}_{t,i}) + \mathcal{L}_{\text{insert}}(\theta; \tilde{\boldsymbol{x}}_{t,i})) \right]$
    Update θ with $\nabla_\theta \mathcal{L}_{\text{AJD}}(\theta)$
  end for
  for `epoch` in $1, \ldots, N_{\text{alt}}$ **do**
    ▷ *Fine-tune planner model with frozen policy* ◁
    $\{\tilde{\boldsymbol{x}}_{t,i}, W^{\bar{\theta},\bar{\phi}}\} \leftarrow \texttt{SampleInterpolant}(\mathcal{B}; R)$
    Predict $f_\theta(\tilde{\boldsymbol{x}}_{t,j}, t), g_\theta(\tilde{\boldsymbol{x}}_{t,j}, t)$
    $\tilde{\boldsymbol{x}}_{s,i}^{\text{unmask}}, \mathcal{M}, \tilde{\boldsymbol{x}}_{s,i}^{\text{insert}}, \mathcal{I} \leftarrow \texttt{OneStepSampler}(\tilde{\boldsymbol{x}}_{t,i}, t, f_\theta(\tilde{\boldsymbol{x}}_{t,j}, t), g_\theta(\tilde{\boldsymbol{x}}_{t,j}, t))$
    Predict insertion quality $\nu_\phi(\tilde{\boldsymbol{x}}_{t,i}^{\text{insert}}, s)$
    $\mathcal{L}_{\text{IQL}}(\phi; \tilde{\boldsymbol{x}}_{t,i}^{\text{insert}}) \leftarrow \sum_{i \in \mathcal{I}} \text{BCE}(\nu_\star^i, \nu_\phi^i(\tilde{\boldsymbol{x}}_{t,i}^{\text{insert}}))$
    Predict unmasking quality $\mu_\phi(\tilde{\boldsymbol{x}}_{t,i}^{\text{unmask}}, s)$
    $\mathcal{L}_{\text{UQL}}(\phi; \tilde{\boldsymbol{x}}_{t,i}^{\text{unmask}}) \leftarrow \sum_{\ell \in \mathcal{M}} \text{BCE}(\mathbf{1}[\tilde{\boldsymbol{x}}_{s,i}^{\text{unmask},\ell} = \boldsymbol{x}_i^{\star,\ell}], \mu_\phi^\ell(\tilde{\boldsymbol{x}}_{s,i}^{\text{unmask}}))$
    $\mathcal{L}_{\text{AJD}}(\phi) \leftarrow \frac{1}{BR} \sum_{(\tilde{\boldsymbol{x}}_{t,i}, W_i^{\bar{\theta},\bar{\phi}}) \sim \mathcal{B}} \left[e^{W^{\bar{\theta},\bar{\phi}}} (\mathcal{L}_{\text{IQL}}(\phi; \tilde{\boldsymbol{x}}_{s,i}^{\text{insert}}) + \mathcal{L}_{\text{UQL}}(\phi; \tilde{\boldsymbol{x}}_{s,i}^{\text{unmask}})) \right]$
    Update ϕ with $\nabla_\phi \mathcal{L}_{\text{AJD}}(\phi)$
  end for
end while

Algorithm 2 `BatchQualitySample`: Sample a batch of sequences from the fine-tuned model and compute the Radon-Nikodym derivative importance weight

Input: Pretrained model $f^{\text{pre}}(\boldsymbol{x}_t, t), g^{\text{pre}}(\boldsymbol{x}_t, t)$, fine-tuned model $f_\theta(\boldsymbol{x}_t, t), g_\theta(\boldsymbol{x}_t, t)$, reward model $r : \mathcal{X} \to \mathbb{R}$, reward scaling α, sampling steps N_{steps}, max length L, tokens to remask N_{remask}
$\Delta t \leftarrow T / N_{\text{steps}}$
$t \leftarrow 0$
for $i = 1$ to N_{steps} **do**
  $\boldsymbol{R}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s) \leftarrow \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot g_\theta(\boldsymbol{x}_t, t)$ ▷ *Predict insertion expectation*
  $\boldsymbol{Q}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s) \leftarrow \frac{\dot{\beta}_t}{1-\beta_t} \cdot f_\theta(\boldsymbol{x}_t, t)$ ▷ *Predict unmasking posterior*
  $\boldsymbol{R}_t^{\text{pre}}(\boldsymbol{x}_t, \boldsymbol{x}_s) \leftarrow \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot g^{\text{pre}}(\boldsymbol{x}_t, t)$ ▷ *Predict pretrained insertion expectation*
  $\boldsymbol{Q}_t^{\text{pre}}(\boldsymbol{x}_t, \boldsymbol{x}_s) \leftarrow \frac{\dot{\beta}_t}{1-\beta_t} \cdot f^{\text{pre}}(\boldsymbol{x}_t, t)$ ▷ *Predict pretrained unmasking posterior*
  $\boldsymbol{x}_s \leftarrow \texttt{SampleCategorical}(\boldsymbol{Q}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s))$ ▷ *Unmasking step*
  if $i < N_{\text{steps}} - 1$ **then**
    $\boldsymbol{x}_s \leftarrow \texttt{ScheduleAwareRemasking}(\boldsymbol{x}_s, \mu_\phi)$
  end if
  $\texttt{log_rnd} \leftarrow \texttt{log_rnd} + \sum_\ell \mathbf{1}[\boldsymbol{x}_s^\ell \neq \boldsymbol{x}_t^\ell] \cdot (\log \boldsymbol{Q}_t^{\text{pre}}(\boldsymbol{x}_s, \boldsymbol{x}_t)[\ell, \boldsymbol{x}_s^\ell] - \log \boldsymbol{Q}_t^\theta(\boldsymbol{x}_s, \boldsymbol{x}_t)[\ell, \boldsymbol{x}_s^\ell])$
  if $i < N_{\text{steps}} - 1$ **then**
    $L_t \leftarrow \text{len}(\boldsymbol{x}_t)$
    $I_t \sim \texttt{Poisson}(\boldsymbol{R}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s) \cdot \Delta t)$ ▷ *Sample number of tokens to insert at each gap*
    $\texttt{total_ext} \leftarrow \sum_\ell L_t^\ell$
    if $\texttt{total_ext} + L_t > L_{\max}$ **then**
      $L_t \leftarrow 0$ ▷ *Do not insert any masks*
    end if
    $L_s \leftarrow L_t + \texttt{total_ext}$
    $\texttt{ext_cum}[L_t + 1] \leftarrow \sum_{j=1}^{\ell} I_t^\ell$ ▷ *Compute cumulative insertions up to position ℓ*
    $\boldsymbol{x}_s' \leftarrow (\text{pad})^{L_{\max}}$ ▷ *Initialize sequence of pad tokens*
    $\boldsymbol{x}_s'[: L_s + 1] \leftarrow \boldsymbol{M}$ ▷ *Initialize L_s tokens to mask*
    for $\ell = 1$ to L_t **do**
      $\boldsymbol{x}_s'[\ell + \texttt{ext_cum}[\ell]] \leftarrow \boldsymbol{x}_s[\ell]$ ▷ *Set unmasked tokens to their new position*
    end for
    $\boldsymbol{x}_s', I_t^\star \leftarrow \texttt{ScheduleAwareDeletion}(\boldsymbol{x}_s', \nu_\phi)$
    $\boldsymbol{x}_s \leftarrow \boldsymbol{x}_s'$
    ▷ *Compute log insertion rates under Poisson* ◁
    $\texttt{log_policy_insert} \leftarrow I_t^\star \log \boldsymbol{R}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s) - \boldsymbol{R}_t^\theta(\boldsymbol{x}_t, \boldsymbol{x}_s)$
    $\texttt{log_pre_insert} \leftarrow I_t^\star \log \boldsymbol{R}_t^{\text{pre}}(\boldsymbol{x}_t, \boldsymbol{x}_s) - \boldsymbol{R}_t^{\text{pre}}(\boldsymbol{x}_t, \boldsymbol{x}_s)$
    $\texttt{log_rnd} \leftarrow \texttt{log_rnd} + \sum_\ell (\texttt{log_pre_insert}[\ell] - \texttt{log_policy_insert}[\ell])$
  end if
  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_s$
  $t \leftarrow t + \Delta t$
end for
$r(\boldsymbol{x}_1) \leftarrow \texttt{RewardFunc}(\texttt{decode}(\boldsymbol{x}_1))$
$\texttt{log_rnd} \leftarrow \texttt{log_rnd} + (r(\boldsymbol{x}_t)/\alpha)$
return $\boldsymbol{x}_1, \texttt{log_rnd}, r(\boldsymbol{x}_1)$

Algorithm 3 `SampleInterpolant`: Partially remasks and deletes tokens in batch

Input: Batch of clean sequences $\{\boldsymbol{x}_{T,i}, W^{\bar{\theta},\bar{\phi}}\}_{i=1}^{B}$, number of replicates R
Repeat each sequence in batch $\{\boldsymbol{x}_{T,j}, W^{\bar{\theta},\bar{\phi}}\}_{j=1}^{B\times R}$
Initialize batch of corrupted sequences $\{\boldsymbol{x}_{t,j}\}_{j=1}^{B\times R}$
for $\boldsymbol{x}_{T,j}$ in $\{\boldsymbol{x}_{T,j}\}_{j=1}^{B\times R}$ **do**
  $t \sim \mathcal{U}(0,1)$
  for $\ell = 1$ to L **do**
    $t_i^\ell \sim \dot{\alpha}_t dt$ ▷ *Sample insertion time*
    $t_u^\ell \sim \mathbf{1}[t \geq t_i^\ell] \frac{\dot{\beta}_t}{1-\beta_{t_i^\ell}} dt$ ▷ *Sample unmasking time*
    if $t < t_i^\ell$ **then**
      $\boldsymbol{x}_{t,j}^\ell \leftarrow$ (empty)] ▷ *Deleted token at time t*
    else if $t_i^\ell \leq t < t_u^\ell$ **then**
      $\boldsymbol{x}_{t,j}^\ell \leftarrow \boldsymbol{M}$ ▷ *Masked token at time t*
    else
      $\boldsymbol{x}_{t,j}^\ell \leftarrow \boldsymbol{x}_{1,j}^\ell$ ▷ *Clean token at time t*
    end if
    Remove intermediate (empty) from $\boldsymbol{x}_{t,j}^\ell$ and replace as pads at end of sequence
  end for
end for
return $\{\boldsymbol{x}_{t,j}, W^{\bar{\theta},\bar{\phi}}\}_{j=1}^{B\times R}$

Algorithm 4 `ScheduleAwareRemasking`: Ensure the number of masked tokens after remasking matches the expected mask count by remasking low-quality tokens.

Input: Draft sequence after unmasking $\tilde{\boldsymbol{x}}_s$, time s, unmasking quality model μ_ϕ, unmasked indices $\mathcal{M}_t$
`exp_num_mask` $\leftarrow$ `interpolant.exp_mask_fraction`$(\tilde{\boldsymbol{x}}_s, s)$
`num_mask` $\leftarrow |\{\ell : \tilde{\boldsymbol{x}}_s^\ell = \boldsymbol{M}\}|$
`mask_to_add` $\leftarrow \max(0,$ `exp_num_mask` $-$ `num_mask`$)$
for $\ell \in \mathcal{M}_t$ **do**
  `remask_scores`$[\ell] \leftarrow -\mu_\phi^\ell(\tilde{\boldsymbol{x}}_s)$ ▷ *lower confidence has higher probability of remask*
end for
$\mathcal{R}_s \leftarrow$ `topk_indices(remask_scores,mask_to_add)` ▷ *set of indices to remask*
for $i \in \mathcal{R}_s$ **do**
  $\tilde{\boldsymbol{x}}_s^\ell \leftarrow \boldsymbol{M}$
end for
return $\tilde{\boldsymbol{x}}_s$

Algorithm 5 `ScheduleAwareDeletion`: Ensure the sequence length after insertion matches the expected length by deleting low-quality insertions

Input: Draft sequence after insertion $\tilde{\boldsymbol{x}}_s$, time s, insertion quality model ν_ϕ, inserted indices $\mathcal{I}_t$, threshold quality μ_{min}
`exp_length` $\leftarrow$ `interpolant.exp_length`$(\tilde{\boldsymbol{x}}_s, s)$
if `len`$(\tilde{\boldsymbol{x}}_s) \geq$ `exp_length` **then**
  `to_delete` $\leftarrow$ `len`$(\tilde{\boldsymbol{x}}_s) -$ `exp_length`
  `insert_scores`$[i] \leftarrow -\nu_\phi^i(\tilde{\boldsymbol{x}}_s)$ ▷ *lower confidence has higher probability of deletion*
  $\mathcal{D}_s \leftarrow$ `topk_indices(insert_scores,to_delete)` ▷ *set of indices to delete*
  $\tilde{\boldsymbol{x}}_s^i \leftarrow$ (empty)
end if
return $\tilde{\boldsymbol{x}}_s, \mathcal{I}_t/\mathcal{D}_s$

J EXAMPLE GENERATIONS

Sampling Steps = 128

A2D2 **answer 4 ✓**

```
Q: A juggler can juggle 16 balls.
Half of
the balls are golf balls, and half of
 the
golf balls are blue. How many blue
golf
balls are there?

Golf balls: 16/2 = 8
Blue: 8/2 = 4 balls
Answer: \boxed{4}
```

Sampling Steps = 256

A2D2 **answer 70 ✓**

```
Q: Gretchen has 110 coins. There are
30
more gold coins than silver coins.
How many
gold coins does Gretchen have?

Let g be the number of gold coins and
 s be
the number of silver coins. We know
that
g + s = 110 and g = s + 30.
Substituting the second equation into
 the
first, we get s + 30 + s = 110
Combining like terms, we get 2s + 30
= 110
Subtracting 30 from both sides, 2s =
80
Dividing both sides by 2, we get s =
40
Substituting back into g = s + 30, we
 get
g = 40 + 30 = 70
Answer: \boxed{70}
```

Sampling Steps = 512

A2D2 **answer 16 ✓**

```
Q: Jenny goes to the florist to buy
some
flowers. Roses cost $2 each and $15
for a
dozen. If she bought 15 roses and
arrived
with five 5 dollar bills and they
only have
quarters for change, how many
quarters does
she leave with?

First find how many individual roses
Jenny
bought: 15 roses - 12 roses = 3 roses
Then multiply by the price per rose:
3 roses * $2/rose = $6
Then add the cost of the dozen roses:
$6 + $15 = $21
Then subtract from the amount Jenny
paid:
$25 - $21 = $4
Then divide the change by the value
of a
quarter: $4 / .25 = 16 quarters
Answer: \boxed{16}
```

Sampling Steps = 1024

A2D2 **answer 40 ✓**

```
Q: Mico and Marco wanted to get to
know
each other. They realized that the
sum of
their ages is 20. What will be the
sum of
their ages in 10 years?

Let's say Mico's age is m.
So Marco's age is 20 - m.
In 10 years, Mico will be m + 10
years old
and Marco will be 20 - m + 10 years
old.
So the sum of their ages will be
m + 10 + (20 - m + 10) = 40.
Answer: \boxed{40}
```

Figure 6: Example GSM8K generations from the any-length model fine-tuned with A2D2 at 128, 256, 512, and 1024 sampling steps. The model is given the problem and generates step-by-step reasoning ending in a `\boxed{}` answer. All four reach the correct answer.

Sampling Steps = 128

A2D2 **reward 2.25**

```
def f(n):
    """ Implement the function f that
        takes n as a
    parameter, and returns a list of
        size n, such that
    the value of the element at index
        i is the factorial
    of i if i is even or the sum of
        numbers from 1 to i
    otherwise. i starts from 1.
    Example: f(5) == [1, 2, 6, 24, 15]
    """
    ret = []
    for i in range(1,n+1):
        if i%2 == 0:
            x = 1
            for j in range(1,i+1): x *=
                j # <-- infilled
            ret += [x]
        else:
            x = 0
            for j in range(1,i+1): x +=
                j
            ret += [x]
    return ret
```

Sampling Steps = 256

A2D2 **reward 2.25**

```
def largest_smallest_integers(lst):
    '''
    Returns a tuple (a, b), where 'a'
        is the largest of
    negative integers, and 'b' is the
        smallest of positive
    integers in a list. If there is no
        negative or positive
    integers, return them as None.
    Examples:
    largest_smallest_integers([2, 4,
        1, 3, 5, 7]) == (None, 1)
    largest_smallest_integers([]) == (
        None, None)
    '''
    smallest = list(filter(lambda x: x
        < 0, lst))
    largest = list(filter(lambda x: x
        > 0, lst)) # <-- infilled
    return (max(smallest) if smallest
        else None,
        min(largest) if largest else
            None)
```

Sampling Steps = 512

A2D2 **reward 2.25**

```
def intersperse(numbers, delimeter):
    """ Insert a number 'delimeter'
        between every two
    consecutive elements of input list
        `numbers`.
    >>> intersperse([1, 2, 3], 4)
    [1, 4, 2, 4, 3]
    """
    if not numbers:
        return []
    result = []
    for n in numbers[:-1]:
        result.append(n)
        result.append(delimeter)
    result.append(numbers[-1]) # <--
        infilled
    return result
```

Sampling Steps = 1024

A2D2 **reward 2.25**

```
def specialFilter(nums):
    """Returns the number of elements
        in the array that
    are greater than 10 and both first
        and last digits of
    a number are odd (1, 3, 5, 7, 9).
    specialFilter([15, -73, 14, -15])
        => 1
    specialFilter([33, -2, -3, 45, 21,
        109]) => 2
    """
    count = 0
    for num in nums:
        if num > 10:
            odd_digits = (1, 3, 5, 7, 9)
            number_as_string = str(num)
                # <-- infilled
            if int(number_as_string[0])
                in odd_digits and \
              int(number_as_string[-1])
                  in odd_digits:
                count += 1
    return count
```

Figure 7: Example HumanEval single-line infilling generations from the any-length model fine-tuned with A2D2 at 128, 256, 512, and 1024 sampling steps. The model is given the function prefix and suffix and infills the line marked `# <- infilled`. All four reconstruct the ground-truth line exactly.