
Constrained Discrete Diffusion

Michael Cardei*
University of Virginia
ntr2rm@virginia.edu

Jacob K. Christopher*
University of Virginia
csk4sr@virginia.edu

Thomas Hartvigsen
University of Virginia
hartvigsen@virginia.edu

Bhavya Kailkhura
Lawrence Livermore National Laboratory
kailkhura1@llnl.gov

Ferdinando Fioretto†
University of Virginia
fioretto@virginia.edu

Abstract

Discrete diffusion models are a class of generative models that construct sequences by progressively denoising samples from a categorical noise distribution. Beyond their rapidly growing ability to generate coherent natural language, these models present a new and important opportunity to enforce sequence-level constraints, a capability that current autoregressive models cannot natively provide. This paper capitalizes on this opportunity by introducing *Constrained Discrete Diffusion* (CDD), a novel integration of differentiable constraint optimization within the diffusion process to ensure adherence to constraints, logic rules, or safety requirements for generated sequences. Unlike conventional text generators that often rely on post-hoc filtering or model retraining for controllable generation, CDD directly imposes constraints into the discrete diffusion sampling process, resulting in a training-free and effective approach. Experiments in toxicity-controlled text generation, property-constrained molecule design, and instruction-constrained text completion demonstrate that CDD achieves *zero constraint violations* in a diverse array of tasks while preserving fluency, novelty, and coherence, while outperforming autoregressive and existing discrete diffusion approaches.

1 Introduction

Language generation, alongside many scientific problems, admits a natural representation as the generation of a discrete sequence from a finite alphabet. Examples range from natural language to molecular SMILES strings and linearized chemical procedures [1–4]. While large language models (LLMs) and related sequence transformers have recently accelerated discovery by proposing candidate sequences with desirable attributes, their autoregressive sampling mechanism produces tokens sequentially, hindering the ability to provide a native mechanism to ensure constrained feasibility. When these constraints are not satisfied, the generated outputs may be unreliable and ineffective in real-world applications. For example, a single out-of-vocabulary atom in a SMILES string can render a synthesized compound meaningless or even dangerous; similarly an overlooked volume unit in an autonomous laboratory protocol can trigger unsafe reactions.

To limit such risks, LLMs are often deployed with a variety of guardrails. These include soft alignment through reinforcement learning from human feedback (RLHF), rejection sampling, or heuristic post-processing [3, 5]. However, these methods do not offer provable compliance with logical formulas, stoichiometric rules, or regulatory thresholds. This is exacerbated by the autoregressive nature of

*Equal contribution.

†Contact author.

common language models, which generate sequences one token at a time, making it difficult to enforce constraints at the sequence level.

In contrast, discrete diffusion models offer a compelling alternative generative mechanism [6–9]. They refine a fully corrupted sequence by iteratively denoising the entire sample, and have shown strong performance across a variety of tasks, including text generation, molecular design, and program synthesis [4, 7, 10]. Since each step exposes a global view of the partial sequence, it creates a natural opportunity to impose sequence-level structure. This work introduces **Constrained Discrete Diffusion (CDD)**, a framework that capitalizes on this property by coupling discrete diffusion with a differentiable projection operator. Given a corrupted sequence at a particular diffusion step, the projection searches for a new candidate that stays close to the model’s current score distribution, preserves entropy-based diversity, and simultaneously satisfies all user-defined constraints before the next denoising update. Because the introduced projection acts only at sampling time, the method is *training-free* and needs neither model retraining nor post hoc filtering. Its stochastic formulation also preserves the generative diversity, which is crucial for exploratory scientific design, as will be shown in the empirical analysis.

Specifically, CDD is evaluated on three representative use cases illustrated in Figure 1: (i) toxicity mitigation with tunable thresholds for safe LLM deployment, (ii) property adherence for molecular sequence generation, and (iii) instruction-following tasks requiring precise symbolic constraints. The experimental results demonstrate *zero threshold violations* for toxicity, up to a *203.4% increase* in novel molecule generation with complete property adherence, and *100% satisfaction* for instruction-following constraints.

Contributions. This paper provides the following contributions:

1. It introduces Constrained Discrete Diffusion (CDD), a novel framework that integrates discrete diffusion models with a differentiable projection operator, enforcing global sequence-level constraints directly within the diffusion sampling process.
2. It formulates this projection operator by solving a Lagrangian dual program at each denoising step, making the constrained optimization tractable and effective for guiding sequence generation.
3. Through five extensive experiments on three different domains, we demonstrate that CDD achieves state-of-the-art constraint adherence (zero violations across all settings) while preserving high sample quality in terms of fluency, novelty, and coherence.

2 Related Work

Recent efforts in controllable text generation have largely focused on constrained decoding, where the output is dynamically restricted to adhere to predetermined syntactic, semantic, or lexical rules. Approaches in this area modify the decoding process to prune the vocabulary, ensuring that only tokens compliant with a formal grammar are considered [11], or employ external modules to filter outputs when direct access to the model’s logits is limited [12]. Other methods have also adapted beam search to encourage the presence or absence of specific words [13, 14]. Although these techniques effectively guide the generation process, they depend on augmented sampling heuristics which encourage, but frequently fail to provide, satisfaction of even simple constraint sets [15]. To address this, other research has implemented mappings from natural language to formal logic allowing verification of factual and logical consistency [16–18] and enforcement of feasibility constraints with external modules [19, 20]. However, these approaches, while effective in their respective domains, often require additional, domain-specific components to ensure that the generated outputs are not only syntactically valid but also adhere to practical or logical requirements.

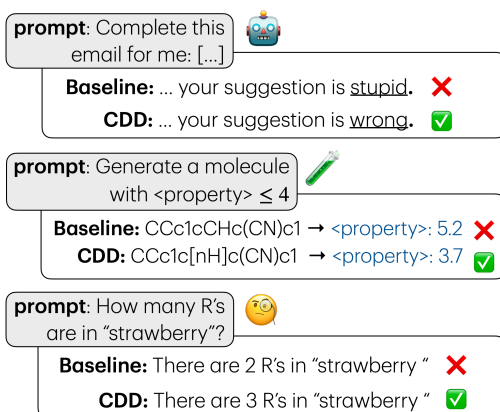


Figure 1: Comparison of **Constrained Discrete Diffusion** and baseline models. CDD imposes constraints without sacrificing fluency or expressiveness.

Other controllable generation approaches rely on energy-based sampling procedures to impose constraints on the outputs. These methods typically define an energy function combining language model likelihood with constraint penalties and leverage algorithms such as Metropolis-Hastings to sample low-energy (i.e., high probability) sequences [21–23]. While such methods are highly effective for encouraging simple properties (e.g., style transfer), they inherit many of the theoretical and practical shortcomings of rejection sampling. In particular, hard constraints are typically enforced only probabilistically, making energy-based approaches less suitable for settings requiring strict constraint satisfaction, as is desirable for the tasks considered in this work.

Finally, several gradient-based sampling frameworks have been proposed to impose constraints on token sequences [24]. Building on this approach, Amini et al. [25] improve generation quality via structured Hamiltonian Monte Carlo. However, these methods lack mechanisms for enforcing hard constraints, and are limited to soft attribute control. Other gradient-based control methods for autoregressive generation, such as PPLM [26], apply conditional steering during decoding to bias generations toward desired attributes; we compare against these methods directly, and show that while it can shift output distributions, it fails to reliably enforce structural or semantic constraints. Guided discrete diffusion methods have also been explored for protein design. For example, [27] applies gradient-based guidance to the denoiser’s hidden states during sampling to steer sequences toward desired property objectives. Compared to these gradient-based guidance methods, our method frames each denoising step as a constrained optimization and enforces feasibility via an augmented-Lagrangian projection, presenting different behavior and formal guarantees.

3 Preliminaries: Discrete Diffusion Models

While diffusion models were originally developed for continuous data, they have recently been extended to discrete domains such as text, enabling non-autoregressive generation [7, 9, 28, 29]. In contrast to autoregressive models which predict tokens one by one, *discrete diffusion* methods generate entire sequences in parallel by first corrupting sequences through a forward noising process and then iteratively reconstructing them with a learned reverse process. *This is a key enabler recognized by this work, which exploits this modus operandi to impose global constraints while simultaneously maintaining high fidelity.*

Let $\mathbf{x}_0 = (\mathbf{x}_0^1, \dots, \mathbf{x}_0^L)$ denote an input sequence of size L , where each token $\mathbf{x}_0^i \in \mathcal{V}$ is represented as a one-hot vector over a vocabulary $\mathcal{V}$ of N distinct tokens. In discrete diffusion models, the forward process produces, at time $t \in [0, T]$, a sequence of values $\mathbf{x}_t \in \mathcal{V}^L$ that parametrizes probability distributions for each token $\mathbf{x}_t^i$ in the sequence. We denote the corresponding sequence of predicted tokens by $\mathbf{x}_t^* = \arg \max(\mathbf{x}_t)$ where the $\arg \max$ operator is applied to every member $\mathbf{x}_t^i$ of the sequence $\mathbf{x}_t$. The diffusion process specifies a *forward transition*, defined as:

$$q(\mathbf{x}_t | \mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \alpha_t \mathbf{x}_0 + (1 - \alpha_t) \nu), \quad (1)$$

where α_t is a schedule that decreases over time, $\text{Cat}(\cdot; p)$ is the categorical distribution over probability vector $p \in \Delta^N$, and ν is a fixed reference distribution that specifies the type of corruption applied to each token. For example, the *Masked Diffusion Language Model* (MDLM) [7] uses a one-hot vector corresponding to the [MASK] token for ν . In contrast, *Uniform Diffusion Language Models* (UDLM) [8], uses ν as the uniform distribution over the vocabulary. Conversely, this instantiation allows tokens to be re-perturbed in later time steps.

In the *reverse process*, $\mathbf{x}_T$ is initialized from ν (e.g., MDLM initializes the probability vectors as one-hots where the weight is fully concentrated on the [MASK] token). For each position i in the sequence, while $\mathbf{x}_t^i = \nu$, the denoiser parameterizes the transitions to clean data, $\mathbf{x}_0$, and the probability of $\mathbf{x}_t^i = \nu$ is discounted as $t \rightarrow 0$. Once $\mathbf{x}_t \neq \nu$, the probability vector transitions into a one-hot vector, concentrating all weight exclusively on the predicted token.

$$q(\mathbf{x}_s | \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \text{Cat}(\mathbf{x}_s; \mathbf{x}_t), & \text{if } \mathbf{x}_t \neq \nu, \\ \text{Cat}\left(\mathbf{x}_s; \frac{(1-\alpha_s)\nu + (\alpha_s - \alpha_t)\mathbf{x}_\theta(\mathbf{x}_t, t)}{1-\alpha_t}\right), & \text{if } \mathbf{x}_t = \nu. \end{cases} \quad (2)$$

Equation (2) formalizes this reverse process parameterization, where $\mathbf{x}_\theta(\mathbf{x}_t, t)$ denotes the trained denoiser and s is the subsequent timestep in the continuous time reverse process and $0 \leq s < t \leq T$. Despite different corruption kernels (absorbing [MASK] [7] vs. uniform [8]), both samplers expose the full token–probability tensor $\mathbf{x}_t$ for all positions at every reverse step.

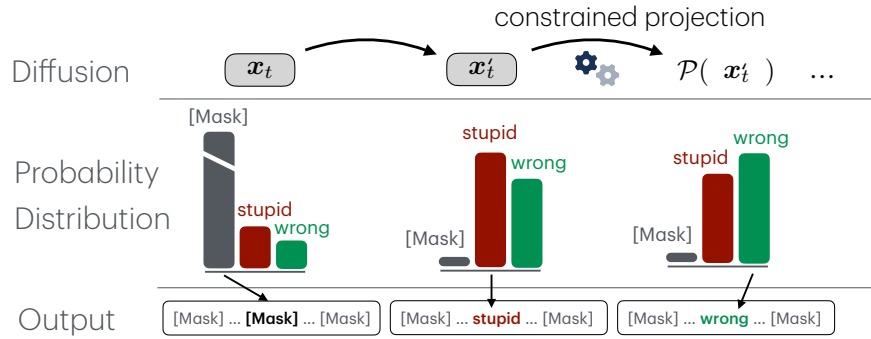


Figure 2: Illustration of CDD’s projection step embedded throughout the sampling process.

4 Constrained Discrete Diffusion (CDD)

4.1 Projected diffusion sampling.

We begin by introducing a perspective of the discrete diffusion reverse process that motivates our approach. Prior work has shown that *continuous* score-based diffusion sampling processes can be framed as an sequential optimization procedure [30]. Score-based parameterizations enable this framing as the model learns to predict the gradients of the data density function, $\mathbf{x}_\theta(\mathbf{x}_t, t) \approx \nabla \log q_t(\mathbf{x}_t | \mathbf{x}_0)$ through Score Matching, and these gradients can then be applied to optimize $\mathbf{x}_t$ to the data distribution $q_t(\mathbf{x}_t)$. This is typically conducted through a Langevin dynamics algorithms [31], which is used either directly [32] or through predictor-corrector Euler-discretizations [33]. In both cases, a series of M steps of Langevin dynamics are taken for a fixed distribution $q_t(\mathbf{x}_t)$:

$$\mathbf{x}'_{t(\ell)} \xleftarrow{\text{update}(\ell)} \mathbf{x}_{t(\ell)} + \gamma_t \nabla_{\mathbf{x}_{t(\ell)}} \log q_t(\mathbf{x}_{t(\ell)} | \mathbf{x}_0) + \sqrt{2\gamma_t} \epsilon \quad (\text{for } \ell = 1 \dots M), \quad (3)$$

where γ_t is the step size, ϵ is added noise, and $\log q_t(\mathbf{x}_t)$ is the density function of the *learned* distribution at time step t . Note that while *annealing* is used to improve convergence, it is applied only after every M iterations. At that point, the sample is updated ($\mathbf{x}_{t(M)} \rightarrow \mathbf{x}_{t-1(1)}$) and the model transitions to the next distribution, $q_{t-1}(\mathbf{x}_{t-1} | \mathbf{x}_0)$, however, the step size and the distribution $q_t(\mathbf{x}_t | \mathbf{x}_0)$ remain stationary throughout these M iterations.

Discrete diffusion models can leverage a discrete generalization of the score function, referred to as the *Concrete score*, to approximate the gradient of the probability density function [7, 9, 34]. Concrete Score Matching provides an approach mirroring continuous Score Matching in that the estimated gradients of the probability density function are used to guide the sample to high density regions of the target distribution. While not always explicitly framed as Concrete Score Matching [9, 34], the denoiser often implicitly models the score function, as supported by theoretical results in [7] demonstrating its equivalence in simplified formulations ($\nabla \log q_t(\mathbf{x}_t | \mathbf{x}_0) \approx \langle \mathbf{x}_\theta(\mathbf{x}_t, t), \mathbf{y} \rangle \quad \forall \mathbf{y} = 1, \dots, N$, in this case). This enables the use of Langevin-based samplers for discrete diffusion, as commonly employed, e.g., in [34].

Effectively, under some regularity conditions, Langevin dynamics will converge towards a stationary point. As shown by Xu et al., Langevin dynamics acts as an “almost-minimizer” on the optimized function, which in this case will be the negative probability density function, converging within a fixed bound of the optimum. Hence, for each series of steps with a stationary density function, within this fixed bound, the sampling procedure can be framed as the optimization problem,

$$\underset{\mathbf{x}_t}{\text{minimize}} \sum_{t=T}^1 -\log q_t(\mathbf{x}_t | \mathbf{x}_0) \quad \text{s.t. } \mathbf{x}_t \in \mathbf{C}. \quad (4)$$

Here, Equation (4) extends the representation from an unconstrained optimization problem to a constrained version by introducing a constraint set $\mathbf{C}$. However, while iteratively applying the denoiser enables sampling from the posterior distribution, it does not natively incorporate externally defined constraints, or even implicit ones, given the stochastic nature of the process. Previous work

for continuous modalities has proposed enforcing $\mathbf{x}_t \in \mathbf{C}$ by applying a Euclidean projection after each denoising step [30], which is natural for continuous modalities which fall in a real space, but this is misaligned when applied to discrete diffusion which operates over a probability simplex $\mathbf{x}_t \in \Delta^N$.

To address this, we introduce a projection operator that minimizes the *Kullback-Leibler (KL) divergence* between the projected and original probability distributions, as opposed to a Euclidean distance metric. Given the model’s predicted probabilities, the projection is defined as:

$$\mathbf{x}_s = \mathbf{x}_{t(\ell+1)} = \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_{t(\ell)}) \stackrel{\text{def}}{=} \arg \min_{\mathbf{y}} D_{\text{KL}}(\mathbf{x}'_{t(\ell)} \parallel \mathbf{y}) \quad \text{s.t.} \quad \arg \max(\mathbf{y}) \in \mathbf{C}. \quad (5)$$

The integration of this projection operator ensures that $\mathbf{x}_s$ is the “nearest sample” that falls within the feasible subdistribution according to a distance metric defined over probability distributions. This ensures the denoising trajectory remains within the allowable set when $\mathbf{C}$ is convex, enabling effective navigation along its boundary or interior toward an optimal solution (illustrated in Figure 2). Next, we show how this projection operator can be formulated as a subproblem within the sampling procedure and efficiently solved using gradient-based methods. Moreover, while convergence guarantees are available for convex constraint sets, as discussed below, Section 5 demonstrates how this technique can effectively handles highly non-linear constraints, including toxicity mitigation, molecular generation, and instruction following, achieving zero violations across all cases.

Importantly, while state transitions are imposed on the token probability distributions, constraint satisfaction is evaluated on the *decoded sequence*. Indeed, the constraints are formulated such that the $\arg \max$ of the projected sequence $\mathbf{y}$ (referred to as $\mathbf{y}^*$), must satisfy sentence level criteria $\mathbf{y}^* \in \mathbf{C}$.³ However, this use of a non-differentiable $\arg \max$ operation also poses challenges to the resolution of the projection, which relies on gradient-based optimization.

4.2 Differentiable projection.

To address the non-differentiability of the $\arg \max$ function, we apply a Gumbel-Softmax relaxation $\tilde{\phi}$, which approximates discrete token probabilities with continuous values [36] as

$$\tilde{\phi}(\mathbf{x}_t^i)(v) = \frac{\exp\left(\frac{\log \mathbf{x}_t^i(v) + \xi_v}{T_{\text{sample}}}\right)}{\sum_{v'=1}^N \exp\left(\frac{\log \mathbf{x}_t^i(v') + \xi_{v'}}{T_{\text{sample}}}\right)}, \quad (6)$$

where $\mathbf{x}_t^i(v)$ is the probability of token v in the vocabulary, ξ_v is drawn from the $\text{Gumbel}(0, 1)$ ¹ distribution for token v , and $T_{\text{sample}} > 0$ is the temperature parameter controlling the smoothness of the output. This enables gradient propagation during the projection step, while closely approximating the discrete $\arg \max$ operation.

4.3 Augmented Lagrangian projection.

Consider a generic constraint on a sequence $\mathbf{x}$ defined via a measurable property given by a function $g_i(\mathbf{x})$. For instance, $g_i(\cdot) : \mathbb{R}^{L \times N} \rightarrow \mathbb{R}^+$ might evaluate sentence toxicity (see Section 5.1), count the number of words occurring in a string (see Section 5.3), or even serve as a black-box function computed by an external routine, as in our molecular sequence generation application (see Section 5.2). To guide the projection operations, we require a measure of constraint violation that can later inform the parametrization of our projection update rule; for ease of presentation, we express the constraint in the form $g_i(\mathbf{x}) < \tau_i$, where $\tau_i \geq 0$ represents an acceptable threshold that must not be exceeded. To quantify by how much a given sequence violates the constraint, we define

$$\Delta g_i(\tilde{\phi}(\mathbf{x}_t)) = \max\left(0, g_i(\tilde{\phi}(\mathbf{x}_t)) - \tau_i\right),$$

where $g = (g_1, \dots, g_m)$ can be treated as series of functions corresponding to a series of thresholds $\tau = (\tau_1, \dots, \tau_m)$, and $\Delta g = (\Delta g_1, \dots, \Delta g_m)$ is defined in analogously quantifying m constraints.

In practice, Δg is non-linear, and, thus, to implement Equation (5), we adopt an augmented Lagrangian approach [37]. In augmented Lagrangian methods, the problem constraints are incorporated into the objective of a minimizer via Lagrange multipliers λ and a quadratic penalty term μ . Let $\mathbf{x}'_t$

³We assume a greedy decoding scheme as is standard to current diffusion-based language models.

be the probability distribution after the denoising step at diffusion time t . We introduce a projected distribution $\mathbf{y}$, which is iteratively updated to reduce the constraint violations (measured by the score Δg) while remaining close (in KL-divergence) to $\mathbf{x}'_t$. Concretely the augmented Lagrangian dual function is defined as:

$$\mathcal{L}_{\text{ALM}}(\mathbf{y}, \mathbf{x}'_t; \lambda, \mu) = D_{\text{KL}}(\mathbf{x}'_t \| \mathbf{y}) + \sum_{i=1}^m \lambda_i \Delta g_i(\tilde{\phi}(\mathbf{y})) + \frac{\mu_i}{2} \Delta g_i(\tilde{\phi}(\mathbf{y}))^2$$

where $\lambda = (\lambda_1, \dots, \lambda_m)$ is a non-negative Lagrange multiplier and $\mu = (\mu_1, \dots, \mu_m)$ is a non-negative quadratic penalty term. When using a Lagrangian function, the primal optimization problem becomes

$$\arg \min_{\mathbf{y}} \mathcal{L}_{\text{ALM}}(\mathbf{y}, \mathbf{x}'_t; \lambda, \mu),$$

and is a lower bound of the original projection operator (5) by weak duality [37]. To obtain the strongest Lagrangian relaxation of the projection, the *Lagrangian dual* can be used to find the best Lagrangian multipliers λ and penalty terms μ , i.e.,

$$\arg \max_{\lambda, \mu} \left(\arg \min_{\mathbf{y}} (\mathcal{L}_{\text{ALM}}(\mathbf{y}, \mathbf{x}'_t; \lambda, \mu)) \right). \quad (7)$$

In practice, the Lagrangian dual is a strong approximation of the original problem (our projection into the constraint space).

The optimization of (7) proceeds iteratively, following a gradient-based update on $\mathbf{y}$ while dynamically adjusting the Lagrange multiplier λ and penalty coefficient μ . Specifically, we perform the following updates [38]:

$$\mathbf{y} \leftarrow \mathbf{y} - \eta \nabla_{\mathbf{y}} \mathcal{L}_{\text{ALM}}(\mathbf{y}, \mathbf{x}'_t; \lambda, \mu) \quad (8a)$$

$$\lambda \leftarrow \lambda + \mu \Delta g(\mathbf{y}^*), \quad (8b)$$

$$\mu \leftarrow \min(\alpha \mu, \mu_{\max}), \quad (8c)$$

where η is the gradient step size, $\alpha > 1$ is a scaling factor that progressively increases μ over iterations, and $\mu_{\max}$ is an upper bound on the penalty term. These updates drive $\mathbf{y}$ as close as possible to satisfy $\Delta g(\tilde{\phi}(\mathbf{y}^*)) \leq \tau$ while also ensuring it remains close to the original denoised distribution $\mathbf{x}'_t$. Pseudocode is provided for our implementation in Algorithm 1.

Algorithm 1: Augmented Lagrangian

Init: $\lambda, \mu, \eta, \alpha, \delta$;

Input: probability distribution $\mathbf{x}'_t$

$\mathbf{y} \leftarrow \mathbf{x}'_t$;

while $\Delta g(\mathbf{y}^*) < \delta$ **do**

for $j \leftarrow 1$ **to** max_inner_iter **do**

$\mathcal{L}_{\text{KL}} \leftarrow D_{\text{KL}}(\mathbf{x}'_t \| \mathbf{y})$

$\mathcal{L}_{\text{viol}} \leftarrow \sum_{i=1}^m \lambda_i \Delta g_i(\tilde{\phi}(\mathbf{y})) + \frac{\mu_i}{2} \Delta g_i(\tilde{\phi}(\mathbf{y}))^2$

$\mathcal{L}_{\text{ALM}} \leftarrow \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{viol}}$

$\mathbf{y} \leftarrow \mathbf{y} - \eta \nabla_{\mathbf{y}} \mathcal{L}_{\text{ALM}}$

$\lambda \leftarrow \lambda + \mu \Delta g(\mathbf{y}^*)$

$\mu \leftarrow \min(\alpha \mu, \mu_{\max})$

$\mathbf{x}_s \leftarrow \mathbf{y}$;

Output: $\mathbf{x}_s$

To assess the robustness of this projection formulation, we performed an ablation study (Appendix C) examining the sensitivity of the Lagrangian hyper-parameters. Across the full grid of settings, *the Lagrangian relaxation converges to feasible solutions in all our test cases*.

As shown in the next section, there is a high degree of flexibility in how these constraints can be implemented. For instance, they can be implemented as surrogate models (e.g., a classifier) that can be used to provide a continuous score, allowing for smooth gradient-base updates, as shown above.

4.4 Theoretical justification.

The next result shows that the constrained reverse diffusion process converges to samples within the feasible region $\mathcal{C}$ while also keeping their distribution close to the data manifold, thus ensuring that the generated samples are both valid and realistic. Let $D_{\text{KL}}(\mathbf{x}_t, \mathcal{C}) = \inf_{\mathbf{y} \in \mathcal{C}} D_{\text{KL}}(\mathbf{y} \| \mathbf{x}_t)$ denote the KL divergence from $\mathbf{x}_t$ to the set $\mathcal{C}$.

Theorem 4.1 (Convergence of CDD). *Let $\mathcal{C}$ be non-empty and β -prox-regular in the sense of [39, Def. 13.27], and the score network satisfy $\|\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\| \leq G$ (a standard consequence of the bounded-data domain after normalization). Then, for positive step sizes $\gamma_t \leq \frac{1}{2G^2} \beta$, the following inequality holds for the distance to the feasible set $\mathcal{C}$:*

$$D_{\text{KL}}(\mathbf{x}'_s, \mathcal{C}) \leq (1 - \alpha_t) D_{\text{KL}}(\mathbf{x}'_t, \mathcal{C}) + \alpha_{t+1}^2 G^2, \quad (\text{non-asymptotic feasibility})$$

where α_t is proportional to the discrete Langevin step size γ_t and G bounds the score norm.

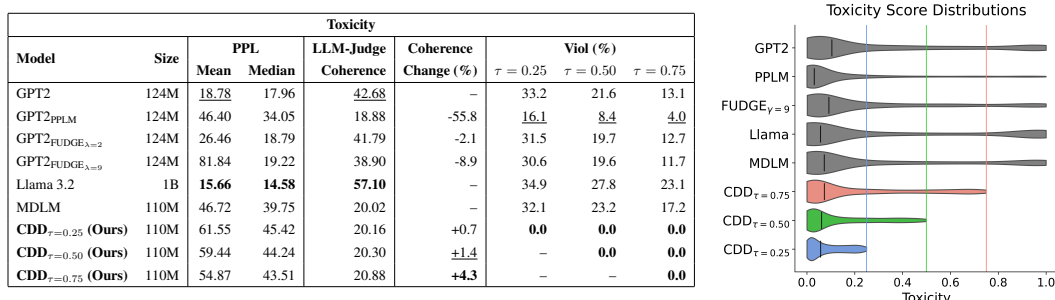


Figure 3: Results across different toxicity thresholds, PPL percentiles, LLM-Judge coherence, and coherence degradation levels. PPL is evaluated with GPT-2-XL, and coherence metrics are assessed by the LLM-Judge. **Bold** and underlined values denote the best and second-best, respectively.

Theorem 4.1 shows that the distance to the feasible set $\mathcal{C}$ decreases at a rate of $1 - \alpha_t$ at each step (up to an additive $\alpha_t^2 G^2$ noise). This implies ε -feasibility after $\mathcal{O}(\alpha_{\min}^{-1})$ steps, with $\alpha_{\min} \propto \min_t \gamma_t$, and a cumulative KL drift within $\mathcal{O}(\sum_t \alpha_t)$. The theorem assumes β -prox-regularity, which provides a relaxation of typical convexity assumptions, and implies that for each viable point and normal direction, small perturbations still project uniquely and smoothly back to $\mathcal{C}$. Theorem proofs are provided in Appendix F.

5 Experiments

To empirically demonstrate the advantage provided by CDD, we compare our method against similarly sized autoregressive and diffusion-based language models. Specifically, we benchmark our performance on constrained text generation against autoregressive baselines: GPT-2 and Llama 3.2 [40, 41], and discrete diffusion baselines MDLM and UDLM [7, 8]. We intentionally select models of similar size to, and up to one magnitude larger than, our base discrete diffusion model, noting that we expect the performance of both classes of models to scale similarly with size. As demonstrated by Schiff et al. [8], MDLM outperforms UDLM on the natural language tasks discussed in Sections 5.1 and 5.3. Consequently, we compare to both models in Section 5.2 and solely MDLM for the following natural language tasks. We use UDLM as the base discrete diffusion model for Section 5.2, and MDLM in Sections 5.1 and 5.3. For each application, CDD uses configurations as described in [7, 8] unless otherwise specified. Additional experiment details are available in Appendix B.

5.1 Natural Language Toxicity Mitigation

While LLMs have demonstrated remarkable capabilities in generating human like text, a byproduct of data driven generation is the inadvertent creation of harmful, offensive and dangerous text. Despite undergoing alignment finetuning, current state-of-the-art models still frequently generate harmful outputs. While post-processing methods can be effective in moderating generations, they can be circumvented by adversarial or malicious prompts, resulting in toxic output generation. In this experiment, we generate outputs conditioned on prompts from the RealToxicityPrompts dataset [42]. We select prefixes with toxicity score above 0.5, testing how these models perform on toxic inputs, and perplexity below 30.0, avoiding incoherent generations. To model the constraints, GPT-Neo [43] is finetuned on the Jigsaw Toxic Comment Classification Challenge⁴. We evaluate on 1,000 samples, and discuss ALM hyper-parameters in Appendix C, runtime in Appendix D, and diversity as measured by entropy [10] in Appendix E.

Figure 3 presents a comprehensive evaluation on model performance for toxicity mitigation, where the objective is to have the generated output satisfy some toxicity threshold, based on a surrogate model, while maintaining competitive perplexity and coherence. We report perplexity (PPL), coherence as determined by a LLM judge, and violation rates at toxicity thresholds $\tau = 0.25$, $\tau = 0.50$, and $\tau = 0.75$ across different model architectures and baselines.

⁴<https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

Among the baselines, GPT2 (124M) and Llama (1B) achieve the lowest perplexity and highest coherence score, however, they frequently produce toxicity content resulting in high violations. Although PPLM and FUDGE (124M) [15, 26] are able to decrease the violations for all toxicity thresholds, they are unable to consistently prevent toxic generations while suffering from an increase in perplexity and a decrease in coherence. MDLM (110M) exhibits higher perplexity than GPT2 and lower coherence, and similarly shares a non-negligible amount of toxicity constraint violations. In contrast, CDD provides **perfect constraint satisfaction** at each toxicity thresholds, maintaining comparable perplexity scores to its base diffusion language model and other baselines. Additionally, while constraint imposition degrades the coherence of the controllable generation baselines, CDD observes an increase in coherence as evaluated by the LLM-Judge. *These findings are important as they show how, using projection-based control strategies, it is possible to achieve complete toxicity constraint satisfaction at desired thresholds.*

5.2 Molecular Generation

Next, we show the ability of CDD to impose constraints on a scientific task, specifically in the domain of molecular generation. In this experiment we generate SMILES strings [44], a linear representation of a molecule's structure with a vocabulary consisting of a limited vocabulary of forty symbols and a grammar dictating molecular validity. Recent advances in generative modeling have enabled the design of molecular sequences by leveraging techniques from natural language processing. Despite their impressive ability to optimize for specific chemical properties, these models often generate molecules that fall short of practical requirements, either by producing compounds that are difficult to synthesize or by closely mimicking existing structures. Furthermore, generated molecules often fail to maximize qualitative properties such as drug-likeness (QED) [45] that are critical to the practicality of the generations and used as a qualitative metric for the experiments discussed in this section.

To address this challenge, our framework incorporates two distinct constraint mechanisms: **(1) synthetic feasibility**, which ensures that generated molecules can be synthesized in a laboratory setting [46], and **(2) novelty**, which guarantees that the generated molecules are not already present in training datasets. For each constraint, we implement a dedicated projection operator, described in B, that imposes hard limits on the generation process, resulting in final outputs that adhere precisely to our desired standards of synthetic accessibility or novelty.

Synthetic accessibility constraints. Synthetic accessibility is commonly assessed using black-box, non-differentiable functions. To enable gradient-based optimization, we train a surrogate model on the QM9 dataset [47], where training labels are derived from RDKit's synthetic accessibility score [48]. This allows us to approximate a differentiable function $g(\cdot)$ for synthetic accessibility. While our molecular generation framework employs this surrogate model to optimize for synthetic accessibility during training, the actual assessment of accessibility violations is conducted by a separate, black-box external model. This external evaluation rigorously measures the degree to which generated molecules comply with the synthetic accessibility criteria, using a series of thresholds ($\tau = 3.0, 3.5, 4.0$, and 4.5). For the discrete diffusion baselines, we adapt the guidance mechanisms provided by [8] for QED by retraining on synthetic accessibility labels for Classifier-Based Guidance (CBG) and Classifier-Free Guidance (CFG). A similar adaptation is applied for the autoregressive baselines employing FUDGE guidance [15]. The results, summarized in Figure 4 (left), confirm that CDD achieves perfect compliance across all thresholds. This 0% violation rate is achieved, all while generating a competitive number of valid molecules and exhibiting the highest drug-likeness scores.

Novelty constraints. Novelty in molecule generation refers to the model's ability to produce new chemical structures that were not explicitly contained in its training set. In the context of generative modeling for drug design or other chemistry applications, novelty is often an important objective for the reasons of chemical space exploration and practical relevance. In this experiment, we generate SMILES strings constraining all valid generations to be novel. The novelty constraint is enforced at every denoising step. If the current candidate sequence already exists in an external database, a projection operator minimally perturbs its token-probability vector to yield an unseen molecule, thereby approximating a gradient step through the otherwise non-differentiable novelty indicator. Specifically, the novelty projection operator is a best-first traversal of the token-probability space: each token flip incurs a cost equal to its probability gap from the argmax, a priority queue retrieves the lowest-cost unseen sequence, and we then renormalize and cache it to prevent duplication. As it selects the sequence with the minimal cumulative flip cost, our distance metric over sequences, the

Molecules (Synthetic Accessibility)							
Model	Valid	Novel	QED	Viol (%)			
				$\tau = 3.0$	$\tau = 3.5$	$\tau = 4.0$	$\tau = 4.5$
AR	1023	0	0.46	91.6	78.4	62.3	42.0
AR _{FUDGE} $\gamma=7$	925	6	0.48	11.1	9.8	9.6	9.2
MDLM	596	20	0.45	85.9	73.7	61.1	44.0
MDLM _{D-CFG} $\gamma=3$	772	3	0.41	87.8	73.9	54.2	22.5
MDLM _{D-CBG} $\gamma=3$	436	14	0.37	50.5	48.6	46.1	44.7
UDLM	895	21	0.47	89.4	88.0	58.1	37.8
UDLM _{D-CFG} $\gamma=5$	850	18	0.47	80.6	58.6	35.9	13.9
UDLM _{D-CBG} $\gamma=10$	896	28	0.47	90.1	77.8	58.6	37.7
CDD_{$\tau=3.0$} (Ours)	353	36	0.63	0.0	0.0	0.0	0.0
CDD_{$\tau=3.5$} (Ours)	863	22	<u>0.62</u>	—	0.0	0.0	0.0
CDD_{$\tau=4.0$} (Ours)	936	31	0.61	—	—	0.0	0.0
CDD_{$\tau=4.5$} (Ours)	938	<u>33</u>	0.58	—	—	—	0.0

Molecules (Novelty)				
Model	Size	Valid & Novel	QED	Viol (%)
No Guidance				
AR	92M	11	0.41	98.93
MDLM	92M	271	<u>0.45</u>	<u>54.53</u>
UDLM	92M	<u>345</u>	0.46	61.45
CDD (Ours)	92M	511	<u>0.45</u>	0.00
CFG				
AR _{D-CFG} $\gamma=3^\dagger$	92M	79	<u>0.60</u>	91.61
MDLM _{D-CFG} $\gamma=3^\dagger$	92M	<u>96</u>	<u>0.60</u>	<u>69.82</u>
UDLM _{D-CFG} $\gamma=5^\dagger$	92M	64	0.62	93.69
CDD_{D-CFG} $\gamma=5$ (Ours)	92M	251	<u>0.60</u>	0.00
CBG				
AR _{FUDGE} $\gamma=7^\dagger$	92M	53	0.61	94.28
MDLM _{D-CBG} $\gamma=3^\dagger$	92M	<u>117</u>	0.58	<u>72.08</u>
UDLM _{D-CBG} $\gamma=10^\dagger$	92M	64	0.61	93.59
CDD_{D-CBG} $\gamma=10$ (Ours)	92M	355	0.59	0.00

Figure 4: **Left:** Results for synthetic accessibility constrained molecule generation constraints. QED and constraint violations are reported for only valid molecules, and novel molecules must be valid and have no violation ($\tau \leq 3.0$). **Right:** Results for novelty projection with and without QED guidance. Violation represents percentage of valid, but not novel molecule generations. QED is reported for only novel molecules. Results denoted with † are as reported by Schiff et al. [8]. **Bold** and underlined values mark the best and second-best, respectively.

procedure is mathematically equivalent to projecting the distribution onto the set of novel sequences. Repeating this operation throughout the diffusion trajectory guarantees that only new compounds are emitted while preserving the exploratory capacity of the discrete diffusion model.

We compare against baselines with no QED guidance alongside CFG and CBG QED guidance [8] to evaluate the impact of our novelty constraint. The results are shown in the right side of Figure 4. Notice how the CBG setting yields a 203.4% increase in novel molecule generation, while the CFG setting shows a 161.4% increase. Even without guidance, the method still produces 48.1% more novel molecules, with only a minimal reduction in the QED score. *These results are significant because they demonstrate that CDD can effectively generate novel molecules while maintaining a high QED score, which is crucial for many scientific tasks.*

5.3 Instruction Following

Finally, the last application focuses on satisfying user-specified symbolic constraints, a task that many autoregressive models struggle with. We evaluate CDD against such models using two types of constraints. First, we assess the ability of the tested models to accurately answer how many letters are in a word, such as ‘How many R’s are in *Strawberry*?’. We additionally test lexical constraints based of the COLLIE dataset [49], prompting the models to generate word-indexed constrained sentences, such as ‘Generate a sentence where the third-to-last word is *mankind*’. For each constraint, the projection operator imposes *hard constraints* on the generation, pushing the samples back onto the constraint set.

	Model	Size	PPL	LLM-Judge	Viol (%)
Counting	GPT-2 (zero-shot)	124M	19.2 $\pm$ 0.8	57.8	100
	GPT-2 (instruction-tuned)	124M	23.4 $\pm$ 1.3	70.9	<u>5.0</u>
	Llama 3.2 (zero-shot)	1B	34.3 $\pm$ 3.8	<u>79.5</u>	100
	Llama 3.2 (instruction-tuned)	1B	30.7 $\pm$ 2.9	91.8	9.1
	MDLM	169M	16.9 $\pm$ 1.0	61.2	54.5
	CDD (Ours)	169M	<u>17.9 $\pm$ 1.1</u>	60.4	0.0
Lexical	GPT-2 (zero-shot)	124M	26.6 $\pm$ 4.2	28.1	97.6
	GPT-2 (instruction-tuned)	124M	62.2 $\pm$ 2.9	15.4	96.4
	Llama 3.2 (zero-shot)	1B	11.3 $\pm$ 2.0	30.7	97.2
	Llama 3.2 (instruction-tuned)	1B	66.5 $\pm$ 6.1	22.9	<u>93.6</u>
	MDLM	169M	<u>21.9 $\pm$ 2.0</u>	<u>34.8</u>	97.5
	CDD (Ours)	169M	26.6 $\pm$ 4.1	65.1	0.0

Table 1: PPL is assessed using GPT2. **Bold** and underlined mark the best and second-best, respectively.

Counting constraints. As illustrated in the top of Table 1, unconstrained GPT-2 and Llama 3.2 fail to satisfy any letter-count requirement. Instruction tuning helps, reducing violation rates to 5.0% (GPT-2) and 9.1% (Llama 3.2), but still leaves a notable error margin. Alternatively, MDLM achieves the lowest perplexity and second-best coherence, but over half of its outputs violate the constraint. By contrast, *CDD fully eliminates violations* while preserving near-best fluency (17.9 perplexity), demonstrating its provision of strict compliance without sacrificing language quality.

Lexical constraints. The results for lexical constraints are shown in the bottom of Table 1. The zero-shot and instruction-tuned versions of GPT-2 and Llama 3.2 show similarly high violation rates (about 97%), and instruction tuning only minimally improves these figures. MDLM again delivers relatively high fluency (21.9 perplexity, 34.8 coherence) but fails to address the precise positional requirements (97.5% violations). In stark contrast, *CDD ensures perfect adherence* and the best LLM-as-a-Judge score (65.1 coherence), with only a moderate increase in perplexity, thus confirming the effectiveness of its projection mechanism for enforcing hard lexical constraints. *Taken together, these experiments reveal a fundamental limitation of conventional and instruction-tuned language models in scenarios requiring strict constraints. Conversely, discrete diffusion models, when equipped with constraints, can effectively produce outputs that satisfy both counting and lexical requirements, opening new avenues for applications in controlled text generation.*

We include more evaluation and experiments in the Appendix. Specifically, Appendix B details the experimental setup, Appendix C evaluates CDD’s robustness through an extensive hyper-parameter ablation, and Appendix D reports runtime comparisons with the baselines.

6 Limitations

There are two key limitations that are currently faced in the development of constrained discrete models. The first regards the size of currently available discrete diffusion models. While current discrete diffusion models provide comparable performance in fluency and perplexity to similarly sized autoregressive models, including state-of-the-art models such as Llama 3.2 1B, their generative capabilities fall short of large-scale language models. Larger scale, closed-source discrete diffusion models have been developed and boasted impressive performance [29], but larger open-source models are yet to be provided to the research community. It is expected that the release of such larger models will further boost the performance of our proposed methods.

The second key limitation concerns an inherent byproduct of constraint imposition: increased computational overhead. While unconstrained discrete diffusion language models have been shown to provide significant speed-ups over autoregressive language models due to the parallelization of token generation [7, 9, 50], the inclusion of constraints within the generation process may limit this benefit. Nevertheless, this trade-off is often justified, as imposing constraints is essential in scenarios where correctness outweighs generation speed, such as ensuring safety-critical outputs or supporting rigorous scientific exploration. In such contexts, achieving accurate and reliable outcomes is prioritized over computational efficiency.

7 Conclusion

In this paper we presented Constrained Discrete Diffusion (CDD), a novel method for imposing both structured and unstructured constraints during sequence generation. CDD relies on the adoption of discrete diffusion models which enable the consideration of entire sequences when imposing constraints during the generation process. Offering a key departure from current paradigms for controllable text generation, CDD integrates differentiable optimization methods into the diffusion reverse process. This integration, governed by a rigorous mathematical optimization framework, provides precise control for discrete sequence generation. The method is empirically validated on (i) toxicity mitigation and harmful content prevention, (ii) generation of novel molecular sequences that adhere to specific properties for drug discovery, and (iii) imposition of lexical constraints at character and sequence levels, reporting state-of-the-art results for controllable generation in both terms of constraint adherence and quality metrics.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No 2234693. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was partially supported by NSF grants CAREER-2401285, RI-2533631, RI-2334936, DARPA under Contract No. #HR0011252E005, and UVA's National Security Data & Policy Institute, ODNI Contracting Activity #2024-24070100001. The work has also been supported by LLNL under Contract DE-AC52-07NA27344 and by the LLNL-LDRD Program under Project No. 24-ERD-010 and 24-SI-008. This manuscript has been co-authored by Lawrence Livermore National Security, LLC under Contract No. DE-AC52-07NA27344 with the U.S. Department of Energy. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, DOE, DARPA, or DOD.

References

- [1] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *arXiv preprint arXiv:2410.07166*, 2024.
- [2] Jingru Jia, Zehua Yuan, Junhao Pan, Paul E McNamara, and Deming Chen. Decision-making behavior evaluation framework for llms under uncertain context. *arXiv preprint arXiv:2406.05972*, 2024.
- [3] Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *arXiv preprint arXiv:2405.10292*, 2024.
- [4] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*, 2024.
- [5] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [6] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL <https://arxiv.org/abs/2107.03006>.
- [7] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024.
- [8] Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dallaré, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. *arXiv preprint arXiv:2412.10193*, 2024.
- [9] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*, 2024.
- [10] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.
- [11] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*

- Processing*, pages 10932–10952, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.674. URL <https://aclanthology.org/2023.emnlp-main.674>.
- [12] Saibo Geng, Berkay Döner, Chris Wendler, Martin Josifoski, and Robert West. Sketch-guided constrained decoding for boosting blackbox large language models without logit access. *arXiv preprint arXiv:2401.09967*, 2024.
 - [13] Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. *arXiv preprint arXiv:2010.12884*, 2020.
 - [14] Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, et al. Neurologic a* esque decoding: Constrained text generation with lookahead heuristics. *arXiv preprint arXiv:2112.08726*, 2021.
 - [15] Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.276. URL <http://dx.doi.org/10.18653/v1/2021.naacl-main.276>.
 - [16] Nakul Gopalan, Dilip Arumugam, Lawson LS Wong, and Stefanie Tellex. Sequence-to-sequence language grounding of non-markovian task specifications. In *Robotics: Science and Systems*, volume 2018, 2018.
 - [17] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robot Learning*, pages 1084–1110. PMLR, 2023.
 - [18] Hanmeng Liu, Zhiyang Teng, Chaoli Zhang, and Yue Zhang. Logic agent: Enhancing validity with logic rule invocation. *arXiv preprint arXiv:2404.18130*, 2024.
 - [19] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. *arXiv preprint arXiv:2306.06531*, 2023.
 - [20] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.
 - [21] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019.
 - [22] Jarad Forristal, Niloofar Mireshghallah, Greg Durrett, and Taylor Berg-Kirkpatrick. A block metropolis-hastings sampler for controllable energy-based text generation. *arXiv preprint arXiv:2312.04510*, 2023.
 - [23] Daniel Gareev, Thomas Hofmann, Ezhilmathi Krishnasamy, and Tiago Pimentel. Local and global decoding in text generation. *arXiv preprint arXiv:2410.10810*, 2024.
 - [24] Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. Gradient-based constrained sampling from language models. *arXiv preprint arXiv:2205.12558*, 2022.
 - [25] Afra Amini, Li Du, and Ryan Cotterell. Structured voronoi sampling. *Advances in Neural Information Processing Systems*, 36:31689–31716, 2023.
 - [26] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.
 - [27] Nate Gruver, Samuel Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion, 2023. URL <https://arxiv.org/abs/2305.20009>.

- [28] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.
- [29] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.
- [30] Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models, 2024.
- [31] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.
- [32] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- [33] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [34] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Advances in Neural Information Processing Systems*, 35:34532–34545, 2022.
- [35] Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.
- [36] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- [37] Stephen Boyd. Convex optimization. *Cambridge UP*, 2004.
- [38] Ferdinando Fioretto, Pascal Van Hentenryck, Terrence W. K. Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *European Conference on Machine Learning*, volume 12461 of *Lecture Notes in Computer Science*, pages 118–135. Springer, 2020. doi: 10.1007/978-3-030-67670-4_8. URL https://doi.org/10.1007/978-3-030-67670-4_8.
- [39] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.
- [40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.
- [41] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton,

Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kam-badur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shao-liang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collet, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gouget, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyan Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem

- Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- [42] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Real-toxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.
- [43] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58(2), 2021.
- [44] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.
- [45] G Richard Bickerton, Gaia V Paolini, J  r  my Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.
- [46] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.
- [47] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.
- [48] A Patr  cia Bento, Anne Hersey, Eloy F  lix, Greg Landrum, Anna Gaulton, Francis Atkinson, Louisa J Bellis, Marleen De Veij, and Andrew R Leach. An open source chemical structure curation pipeline using rdkit. *Journal of Cheminformatics*, 12:1–16, 2020.
- [49] Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*, 2023.
- [50] Jacob K Christopher, Brian R Bartoldson, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. *arXiv preprint arXiv:2408.05636*, 2024.
- [51] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*, 2022.
- [52] Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. Constrained

- language models yield few-shot semantic parsers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.608. URL <https://aclanthology.org/2021.emnlp-main.608>.
- [53] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL <https://aclanthology.org/2021.emnlp-main.779>.
 - [54] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*, 2022.
 - [55] Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*, 2023.
 - [56] Dongwei Jiang, Marcio Fonseca, and Shay B. Cohen. Leanreasoner: Boosting complex logical reasoning with lean, 2024.
 - [57] Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming, 2023.
 - [58] Nathaniel Weir, Peter Clark, and Benjamin Van Durme. Nellie: A neuro-symbolic inference engine for grounded, compositional, and explainable reasoning, 2023.
 - [59] Ziyi Yang, Shreyas S Raman, Ankit Shah, and Stefanie Tellex. Plug in the safety chip: Enforcing constraints for llm-driven robot agents. *arXiv preprint arXiv:2309.09919*, 2023.
 - [60] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL <https://arxiv.org/abs/2306.05685>.
 - [61] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
 - [62] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Yes, the claims made in the abstract and introduction are reflected and elaborated throughout the paper and appendix.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Yes, please see Appendix 6 for the limitations. Furthermore, thorough ablation studies are available in Appendix C and runtime is discussed in D.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Theoretical results are discussed in 4 with missing proofs in Appendix F.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We provide experimental details in Section 5, description of implementations in Appendix B, and hyperparameter choices in Appendix C. Code will be released following the review process.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use datasets and models that are publicly accessible. Information to faithfully reproduce the experimental results are included in Section 5, and Appendix B and C.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please see Appendix B and C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars and distributional information where applicable.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see Appendix C and D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We followed the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: While positive impacts are discussed in Section 5, there are no negative societal impacts in our work that need to be discussed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not introduce data or models that pose a high risk of misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The creators or original asset owners have been credited in appropriate ways.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: In this work, no new assets are released.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core methodology, scientific rigorousness, or originality of the research does not involve the usage of LLMs

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Expanded Related Work

Constrained Decoding Algorithms. A prominent strategy in controllable text generation is to employ constrained decoding algorithms, which incrementally parse outputs under specified formatting or structure requirements. Such approaches have been extensively used in tasks like code generation [51], semantic parsing [52], SQL queries [53], and grammar-constrained text generation [11]. These methods typically prune the vocabulary at each decoding step to only those tokens that comply with the target representation’s syntax or semantics. For instance, Geng et al. confine the logit distribution to tokens valid within a formal grammar, then employ a *completion engine* to extend partial sequences while maintaining grammatical consistency. Meanwhile, approaches that treat the original language model as a black box resort to external constraints. For example, [12] harnesses a secondary model to parse and restrict outputs, an option that becomes necessary when direct access to the logits is unavailable. However, these workarounds may suffer performance penalties due to their indirect control over the primary model’s probability distribution.

Beyond syntax-focused constraints, lexical constraints have also been extensively studied. Lu et al. [13] devise Neurologic decoding, which encourages the presence or absence of specific words in generated text via a modified beam search. They later improve adherence to these constraints by employing an A* search variant [14]. Similar frameworks combine toxicity classifiers with beam search to steer generation toward more benign text [26, 54]. Although these methods can effectively address a range of constraints, they generally rely on augmented sampling heuristics or additional classifiers rather than natively integrating constraints into the generation process.

Natural Language Verification with Formal Logic. Another line of work has examined mapping natural language into logical representations to ensure factual and logical consistency [16–18]. By encoding linguistic inputs in a formal system, these methods can detect inconsistencies or factual errors and thereby provide a measure of control over generated outputs. For instance, Olausson et al., Jiang et al., and Zhang et al. convert natural language statements into first-order logic (FOL) and leverage theorem proving to verify factual correctness. However, FOL struggles to capture complex temporal or sequential dependencies, limiting its applicability to planning or process-oriented tasks. Similarly, other work uses entailment models to verify that a conclusion logically follows from a set of premises [58], though these methods generally stop short of actively generating text that satisfies domain-specific constraints.

Agent Constraint Satisfaction. In robotics and embodied AI, natural language planning frequently incorporates specialized external modules to handle feasibility constraints. For example, Chen et al. [19] and Lin et al. [20] defer to domain-specific motion planners to generate viable trajectories, embedding constraint satisfaction directly into the underlying optimization routines. Although these solutions can integrate strong, domain-aware constraints, they do so by delegating control to highly specialized components. On the discrete side, Yang et al. [59] achieves constraint satisfaction by running a validation module at each step of plan construction and discarding infeasible actions through rejection sampling. While effective in certain settings, such methods typically require domain-specific expertise and do not uniformly integrate constraint enforcement into the generative model itself.

B Experimental Setup

B.1 Toxicity Mitigation

LLM-as-a-Judge Setup. To quantify the *fluency* and *coherence* of generated continuations we follow the increasingly adopted LLM-as-a-Judge evaluation. This method has been shown to result in similar evaluation as human judgment [60]. Specifically, each post-prefix sentence is scored by the instruction-tuned GEMMA-3-27B-IT model and issued the following prompt:

“You are a language expert evaluating the fluency and coherence of the following AI-generated sentence. . . Give a single integer score between 0 (poor) and 100 (excellent), with no explanation or comments.”

The model returns a single scalar $s \in [0, 100]$ for each sentence. Using an LLM evaluator offers two key advantages: (i) strong correlation with human judgments at negligible annotation cost, and (ii) scalability for thousands of evaluations performed in various experiments.

Surrogate Model. As the surrogate model for toxicity task, we use a GPT-Neo (1.3B) model, adapted for binary classification. We finetune this model on the Jigsaw toxicity dataset which includes multiple toxicity-related labels such as toxic, severe toxic, insult, etc. We consolidate these columns into a single binary target (toxic vs. non-toxic).

PPLM Implementation. As a baseline for toxicity reduction in natural language generation we include the PPLM methodology [26]. PPLM uses a frozen pretrained GPT-2 generator for output steering. At every decoding step the algorithm runs the usual forward pass, takes a few gradient-ascent steps on the cached key/value activations to maximize the score of a small attribute model which in this case is a toxicity classifier trained on the Jigsaw toxicity dataset. It simultaneously keeps the perturbed logits close to the original distribution with a KL penalty and a geometric-mean fusion term. We use the authors implementation and code structure without architectural modifications. The underlying steering mechanism, loss terms and hyper-parameter values are unchanged from the authors’ baseline.

FUDGE Implementation. For another toxicity control baseline we additionally implemented FUDGE [15] on the Jigsaw Toxicity dataset. Following the authors implementation we use an LSTM followed by a linear output layer trained to predict future toxicity from partial prefixes. At inference time we add its scaled log non-toxicity probability to GPT-2’s next-token logits for the top candidates then sample from the re-weighted distribution. While the original implementation of FUDGE fixes the guidance strength $\lambda = 1$, we perform a hyperparameter search and sample from the resulting distribution with $\lambda \in \{1, \dots, 9\}$, displaying the toxicity and perplexity results in Table 2. We use the authors’ original code structure, and base language model.

λ	PPL (Avg)	Coherence	Viol % $\tau = 0.25$	Viol % $\tau = 0.5$	Viol % $\tau = 0.75$
1	20.34	41.64	33.5	20.0	13.1
2	26.46	41.79	31.6	19.7	12.7
3	36.36	40.07	33.1	20.7	13.3
4	43.80	40.37	31.8	21.5	13.3
5	54.37	40.12	31.7	21.6	13.0
6	65.23	39.33	33.1	21.7	13.1
7	75.23	38.76	32.5	20.3	12.9
8	76.51	39.09	32.2	20.8	12.6
9	81.84	38.91	30.6	19.6	11.4

Table 2: FUDGE perplexity, coherence, and violation rates at three thresholds for varying λ .

Diffusion Guidance Implementation. While recent literature [8] has proposed adaptations of classifier-based guidance [61] and classifier-free guidance [62] for discrete diffusion models, these approaches have only been employed for settings with limited vocabulary sizes (e.g., molecule generation where the vocabulary is composed of 40 tokens). For natural language tasks which have much larger vocabulary sizes, for instance, the GPT2 tokenizer is composed of over 50,000 tokens—these techniques have not been applied. Currently, this is a scalability issue, as these approaches require that guidance vectors are computed of size $\mathbb{R}^{L \times N}$, making it much simpler and efficient when operating on a limited vocabulary size. Hence, as there is no prior literature to the extent of our knowledge that has scaled these techniques to natural language tasks, we defer comparison to these methods for the molecular generation setting.

Implementation Details. For the ALM projection, we initialize using the following hyperparameters: $\lambda_{\text{init}} = 0.0$, $\mu_{\text{init}} = 1.0$, $\text{outer_iter}_{\text{max}} = 1000$, $\text{inner_iter}_{\text{max}} = 10$, $\eta = 0.20$. We choose these setting for the toxicity experiment as they result in the lowest perplexity in the Appendix C ablation study. For natural language toxicity reduction details we use a generation length of 100 tokens.

B.2 Molecular Generation

Training Dataset. For our molecule generation experiments, we utilize the QM9 dataset [47], as used by Schiff et al. [8]. This dataset comprises approximately 133,885 small organic molecules,

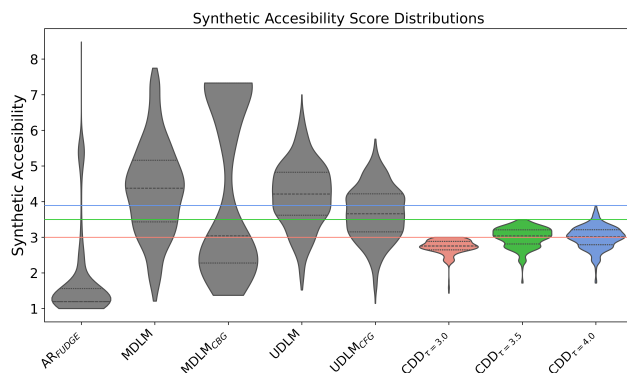


Figure 5: Synthetic Accessibility score distributions for CDD versus competing baselines. Importantly, CDD is the only model that never violates the specified toxicity thresholds.

each encoded as a SMILES string [44], which compactly represents the molecular structure through a sequence of characters. The SMILES representation facilitates discrete modeling by enabling the treatment of molecular generation as a sequence prediction task, making it particularly amenable to discrete diffusion approaches. By leveraging QM9, we can rigorously evaluate the performance of our generative models on tasks that require both the preservation of chemical validity and the precise control of molecular properties, aligning with established protocols in the literature.

B.2.1 Synthetic Accessibility Constraints

Surrogate Model. We train our surrogate model on the QM9 dataset [47] but manually label the training data with RDKit’s synthetic accessibility score [48]. We finetune GPT2 (124M) to act as this surrogate model and directly output a score $s \in [0, 10]$.

FUDGE Implementation. For this setting, we follow the FUDGE implementation provided by [8]. However, while their guidance is trained on QED scores, labeling samples from the QM9 dataset with these scores, we adapt it to label the training data with RDKit’s computation of the synthetic accessibility scores.

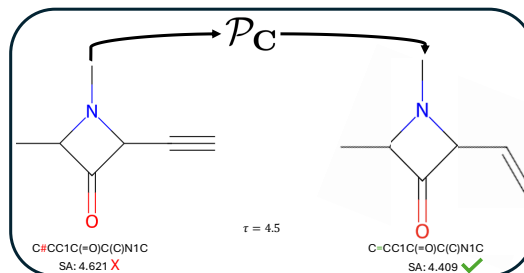


Figure 6: SA projection illustration.

Diffusion Guidance Implementation. Similar to the implementation of FUDGE, we adapt the guidance mechanisms provided by [8] for QED by retraining on synthetic accessibility labels. Otherwise, our implementation mirrors their approach for QED guidance.

ALM Implementation. For the ALM projection, we initialize using the following hyperparameters: $\lambda_{\text{init}} = 0.0$, $\mu_{\text{init}} = 1.0$, $\mu_{\text{max}} = 1000$, $\text{outer_iter}_{\text{max}} = 1000$, $\text{inner_iter}_{\text{max}} = 100$, $\eta = 1.0$.

B.2.2 Novelty Constraints

Novelty Projection Operator. In order to project the molecule sequences into a novel set, we apply a best-first search (BFS) at the probability distribution level. We begin by defining a flip cost for forcing a token to become the new argmax; this cost is the difference between the current top token’s probability and the candidate token’s probability, or zero if the candidate is already top-ranked. To find a novel sequence at minimal total flip cost, we start with an empty sequence and expand it position by position, accumulating flip costs in a priority queue. Once a full sequence is constructed, we decode it and check if it is absent from our existing dataset. If it is indeed novel, we finalize the sequence, shift the probability distribution so that each selected token becomes the definitive argmax, and insert the resulting sequence into the dataset to prevent re-generation. This procedure systematically maintains high-likelihood sequences while avoiding those already present, terminating for each sample as soon as it finds a suitable novel result before proceeding to the next sample.

FUDGE Implementation. For this application, we similarly follow the FUDGE implementation and configuration in [8] for QED guidance.

Diffusion Guidance Implementation. We use classifier based guidance and classifier free guidance for QED as implemented in [8].

B.3 Instruction Following

LLM-as-a-Judge Setup. To quantify the *fluency* and *coherence* of generated continuations we follow the increasingly adopted LLM-as-a-Judge evaluation. This method has been shown to result in similar evaluation as human judgment [60]. Specifically, each post-prefix sentence is scored by the instruction-tuned Gemini-2.0-Flash model and issued the following prompt:

“You are a language expert evaluating the fluency and coherence of the following AI-generated sentence... Give a single integer score between 0 (poor) and 100 (excellent), with no explanation or comments.”

The model returns a single scalar $s \in [0, 100]$ for each sentence. Using an LLM evaluator offers two key advantages: (i) strong correlation with human judgments at negligible annotation cost, and (ii) scalability for thousands of evaluations performed in various experiments.

Discrete Diffusion Base Model. While in Section 5.1, we directly utilize the pretrained MDLM checkpoint provided by [7], for the experiments in Section 5.3, we pretrain and subsequently instruction-tune a slightly larger MDLM model, adopting the Llama 1 tokenizer. Empirical observations indicated that employing a marginally larger model and vocabulary size was essential for optimal performance on instruction-guided tasks. Consequently, we adopt this enhanced model for all subsequent evaluations in this context, including both the base MDLM experiments and the proposed CDD framework. Our instruction tuning is conducted using the ShareGPT dataset, and all pretraining follows the design choices and hyperparameters provided by [7].

B.3.1 Counting Constraints

Instruction-Tuning Dataset. We manually construct a dataset of letter counting examples in an instruction-tuning format. We use random words and select a random character from these words 80% of the time. The other 20% of the time we select a letter at random, which may or may not be contained in the word, to represent cases where the correct answer is zero.

Projection Operator. Our implementation of this projection operator involves integrating an external module to count the number of times a specific letter appears in a given word. First, we leverage the external module to obtain the accurate count, ensuring that the correct numerical value is determined. Then, during output generation, we adjust the probability distribution over the numerical tokens so that the token corresponding to this count is assigned the highest probability, while the probabilities of other numerical tokens are appropriately diminished. This approach guarantees that when the model is queried about the frequency of a letter in a word, it outputs the correct numerical token with the highest weight in the distribution, maintaining consistency and precision in the final answer.

B.3.2 Lexical Constraints

Instruction-Tuning Dataset. We finetune models on a dataset of examples constructed from evaluation of GPT-4 on the COLLIE dataset provided by Yao et al. [49]. As GPT-4 performs well on these examples, we apply knowledge distillation by compiling these generations into an instruction tuning dataset. The dataset can be directly constructed from the logs included in the code base from Yao et al.

Projection Operator. Our implementation of the projection operator begins by parsing the decoded string to locate the position corresponding to the n_{th} word. Because token indices do not necessarily match word indices, owing to the nuances of subword tokenization, we perform an alignment procedure that accurately maps word positions to their respective token indices. Once the index in

the sequence corresponding to the n_{th} word is identified, we apply a closed-form gradient update that adjusts the probability distribution over the vocabulary. This update maximizes the probability of the correct token at the specified position while ensuring that the overall distribution remains valid (i.e., all probabilities sum to one).

C Hyper-parameter Ablation Study

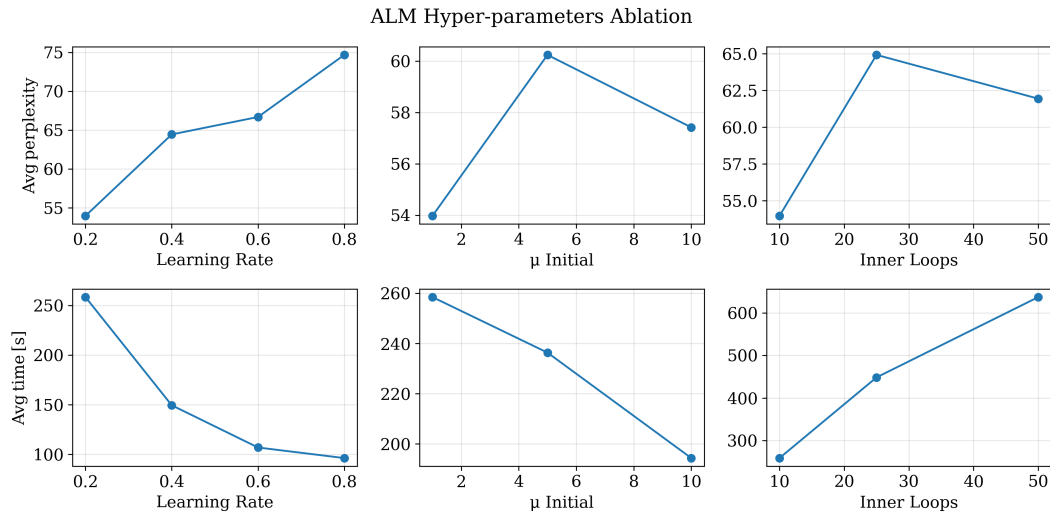


Figure 7: Augmented-Lagrangian Method hyper-parameter ablation study.

To evaluate the robustness of CDD we perform a hyper-parameter sensitivity study for toxicity constraints in natural language generation. We sweep three key Augmented-Lagrangian optimization hyper-parameters: learning-rate, number of inner loop steps, and μ -initialization values. We further examine the impact of projection frequency on constraint generation. In our study we evaluate for constraint satisfaction, perplexity of generated sequences, and runtime. Crucially, for every configuration in the search space the toxicity threshold constraint are **fully satisfied**, indicating that the method remains reliable under substantial hyper-parameter variations. For all test setting we evaluate with the same subset of 200 random samples, and all are run on a single NVIDIA RTX A6000 GPU.

Augmented-Lagrangian Parameters The Augmented-Lagrangian Method lets us express the detoxification goal ($toxicity \leq \tau$) as a smooth, differentiable objective rather than a hard discrete constraint. At every outer iteration ALM augments the loss with (i) a linear Lagrange term that guides the solution toward feasibility and (ii) a quadratic penalty that makes constraint violations increasingly costly. This dual penalty keeps gradients informative even when the constraint is initially violated, yet avoids the numerical instability of a single, very large fixed penalty. The inner optimizer therefore steers the logits toward a region that is both close to the original sentence (via the KL term) and certified non-toxic (via the AL penalty). To assess robustness and sensitivity, we perform a grid search over three key ALM hyper-parameters:

1. **Learning rate** η controls the size of each optimization step on the logits. Higher η accelerates early progress but can overshoot when the penalty weight μ grows; lower η stabilizes training at the cost of more iterations.
2. **Initial penalty weight** μ_0 sets the starting strength of the quadratic term. A small μ_0 allows the optimizer to explore a broader region and preserve fluency, while a large μ_0 enforces the toxicity constraint aggressively from the beginning.
3. **Max inner loops** determines the number of optimization steps taken before updating λ and μ . This value controls the trade-off, where too few steps may lead to an under-optimized solution, whereas too many leads to wasted compute and runtime.

The main trends are summarized in Figure 7 and exhaustively detailed in Table 3.

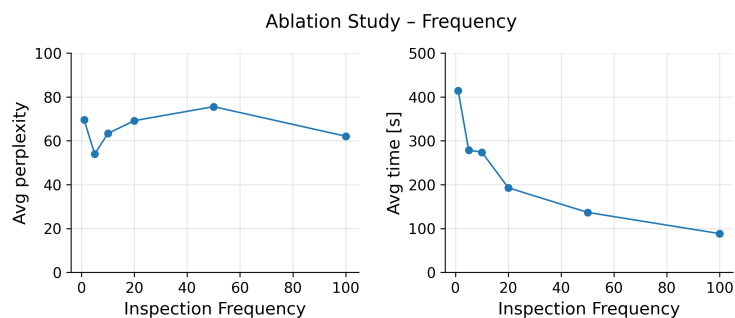


Figure 8: Effect of constraint-checking frequency on runtime and perplexity.

Frequency Ablation We further perform an ablation study with different inspection frequencies. This ablation examines the trade-off between sample quality, constraint satisfaction, and runtime. A more frequent schedule results in consistent adherence at a price of increased compute and runtime. We observe as the inspection frequency increases the average runtime decreases consistently. By analyzing this we demonstrate this method’s robustness and note that for all settings, constraint satisfaction is met. Results are shown in Figure 8.

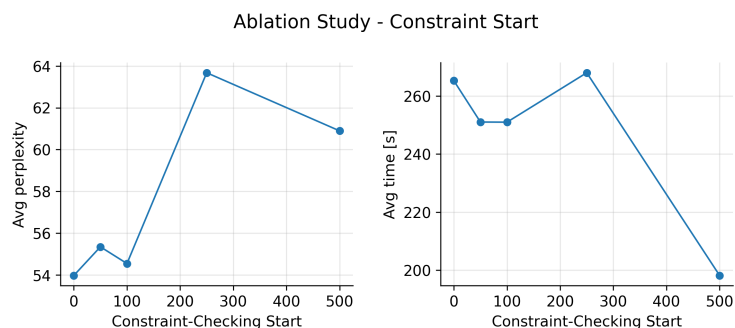


Figure 9: Constraint-checking start analysis on perplexity and runtime.

Constraint Start Ablation We include a constraint-start ablation, where we examine the runtime/perplexity tradeoff by turning the projection operator on only after specific amounts of reverse-diffusion steps. Early enforcements allows the projection to satisfy the constraints at very early iterations at the cost of increased runtime. Notably, across all setting, constraint satisfaction is met for all samples.

D Runtime Analysis

To benchmark the computational overhead of CDD as compared to the baselines, we compare the runtime of our method and the other baselines on the toxicity mitigation task. To normalize for the overhead of base models (MDLM and GPT2), we make two adjustment: we reduce the number of diffusion steps from 1000 to 100 and adjust the generation sequence length to 1024; prior work on discrete diffusion has indicated that these hyperparameters can be used to approximately match the inference time of autoregressive models [9]. Empirically, we find that, while the speed is not identical, these adjustments result in a much more similar tokens-per-second rate for the base models. Practically, MDLM could be accelerated even further by reducing the number of diffusion steps further, but we abstain from this so as to maintain consistency with the baselines. For every system we report the mean generation time, the 25th/50th/75th percentiles, and the slowdown relative to the respective baseline.

ALM hyper-parameters			Perplexity				Wall-clock time (s)				LLM-Judge
LR	μ_{init}	Loops	Mean	25%	50%	75%	Mean	25%	50%	75%	Score
0.20	1	10	53.97	29.91	44.33	66.18	258.52	10.75	12.29	153.54	18.59
0.20	1	25	64.92	28.54	44.34	66.53	448.47	13.15	15.02	374.19	19.46
0.20	1	50	61.95	31.44	45.58	74.66	637.53	13.18	14.82	454.52	20.29
0.20	5	10	60.25	29.15	45.31	71.67	236.39	13.49	15.74	130.67	20.09
0.20	5	25	59.78	28.56	44.38	72.20	343.61	10.54	11.92	192.14	19.64
0.20	5	50	79.71	28.32	44.53	71.44	809.47	10.60	11.93	307.58	19.31
0.20	10	10	57.42	27.69	42.31	75.10	194.37	10.97	13.46	105.79	19.61
0.20	10	25	58.64	27.45	43.73	67.97	636.73	10.56	12.15	126.59	19.25
0.20	10	50	64.90	29.97	45.67	72.46	760.28	11.03	13.86	186.85	20.49
0.40	1	10	64.45	30.65	45.00	74.42	149.41	10.80	12.87	113.80	20.01
0.40	1	25	62.11	33.07	46.49	76.91	223.79	10.55	12.07	200.18	20.44
0.40	1	50	58.95	29.10	44.23	72.99	356.07	10.50	11.52	175.66	20.15
0.40	5	10	58.33	28.29	43.18	66.50	124.73	10.63	11.57	104.85	20.00
0.40	5	25	68.55	29.19	45.34	69.38	213.77	10.66	11.76	204.08	20.11
0.40	5	50	73.02	31.72	45.83	76.92	220.34	10.65	11.74	81.31	20.18
0.40	10	10	69.06	29.22	45.00	73.44	136.86	10.63	12.00	95.29	19.67
0.40	10	25	61.22	30.51	45.34	72.76	193.70	10.77	12.01	123.57	20.11
0.40	10	50	102.30	32.47	46.62	74.88	250.82	10.70	12.44	135.07	19.77
0.60	1	10	66.69	30.22	45.67	74.00	106.81	10.82	11.77	94.56	19.67
0.60	1	25	87.59	32.07	45.77	69.40	196.40	10.77	12.04	170.36	19.10
0.60	1	50	126.00	30.38	46.98	74.85	228.66	10.64	12.02	153.14	19.23
0.60	5	10	65.16	31.96	44.59	72.71	97.96	10.89	12.13	71.26	19.57
0.60	5	25	64.33	29.15	45.80	77.38	144.82	10.76	11.87	144.06	19.34
0.60	5	50	93.99	31.56	47.41	76.91	211.75	10.91	12.14	141.92	19.65
0.60	10	10	65.32	30.79	46.20	77.53	87.94	11.11	12.73	50.27	19.98
0.60	10	25	68.26	30.51	47.78	75.92	158.64	10.76	11.96	70.02	19.27
0.60	10	50	71.86	29.99	44.51	72.47	167.33	10.85	12.17	105.36	19.55
0.80	1	10	74.68	31.81	47.64	78.54	96.14	10.75	11.83	50.56	18.83
0.80	1	25	77.26	30.62	45.77	76.91	158.28	10.77	11.77	115.30	20.57
0.80	1	50	84.82	30.51	45.77	77.38	297.94	11.90	15.24	157.88	19.91
0.80	5	10	68.87	29.05	44.67	70.50	85.59	14.12	16.06	38.68	19.34
0.80	5	25	60.04	26.73	44.05	68.03	129.18	14.84	17.11	64.22	19.60
0.80	5	50	56.15	27.77	44.24	69.18	175.82	15.61	17.76	88.31	19.41
0.80	10	10	74.68	28.95	45.31	74.14	88.95	15.62	17.56	65.28	19.14
0.80	10	25	78.71	29.99	45.67	69.43	146.18	14.99	16.95	105.86	19.41
0.80	10	50	66.70	28.29	44.24	75.66	134.23	10.65	11.74	74.80	19.50

Table 3: Perplexity and wall-clock time statistics, together with LLM-Judge coherence scores, across all ALM hyper-parameter settings.

Toxicity – Runtime						
Model	Size	Runtime (s)				Increase Factor
		Mean	25%	50%	75%	
GPT2	124M	10.31	10.28	10.32	10.36	–
GPT2 _{PPLM}	124M	878.47	857.38	869.69	887.41	85.21
GPT2 _{FUDGE$\lambda=2$}	124M	1387.28	1386.47	1387.00	1387.73	134.56
GPT2 _{FUDGE$\lambda=9$}	124M	1481.94	1479.71	1480.27	1484.47	143.74
Llama 3.2	1B	16.43	16.38	16.44	16.46	–
MDLM	110M	4.59	4.40	4.49	4.74	–
CDD$_{\tau=0.25}$ (Ours)	110M	401.15	9.09	9.67	11.71	87.02
CDD$_{\tau=0.50}$ (Ours)	110M	230.89	9.19	9.62	10.46	<u>50.08</u>
CDD$_{\tau=0.75}$ (Ours)	110M	63.71	9.20	9.56	10.21	13.82

Table 4: Runtime statistics for all evaluated models and respective runtime increase from base model. **Bold** and underlined values mark the best and second-best time increase factor, respectively.

Notably, we find that even without optimizing for runtime (besides selecting specific ALM hyperparameters) CDD provides the *least additional overhead* as compared to other controllable generation methods. Further adjustment (e.g., hot-starting the ALM projection, further reduction in diffusion steps) could be made to increase this margin even further.

E Entropy Analysis

For the toxicity application, we additionally evaluate CDD and its unconstrained base model using entropy. As described in [10], entropy measures the diversity of the generated sequence. For a generated sequence of length L with K distinct tokens, let L_k denote the count of token k , and define the token probabilities $p_k = L_k/L$. The entropy of this distribution is $-\sum_{k=1}^K p_k \log p_k$. Lower entropy indicates that probability mass is concentrated on a few tokens, reflecting a lower diversity, whereas a higher entropy indicates a more uniform token usage and greater generative diversity.

As observed in Table 5, across all thresholds, CDD results in a negligible entropy decrease of 0.59–1.0% compared to the base model. As the decrease is small and within narrow confidence intervals, this validates that CDD preserves generative diversity.

Model	Size	Mean Entropy	95% CI	% Decrease
MDLM	110M	5.577	0.0178	–
CDD $_{\tau=0.25}$	110M	5.521	0.0709	1.00
CDD $_{\tau=0.50}$	110M	5.519	0.0851	1.04
CDD $_{\tau=0.75}$	110M	5.544	0.0709	0.59

Table 5: CDD mean entropy, 95% confidence intervals, and percent decrease from base model at different thresholds for the toxicity application.

F Missing Proofs

Proof of Theorem 4.1

Proof. We begin by proving this bound holds for projected diffusion methods operating in the image space:

$$D_{\text{KL}}(\mathbf{x}'_s, \mathbf{C}) \leq (1 - \alpha_t) D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) + \alpha_{t+1}^2 G^2, \quad (9)$$

For ease of notation, we will denote subsequent timestep in terms of an arbitrary t , such that $\mathbf{x}_{t-1} = \mathbf{x}_s$ and the subsequent timestep after that is denoted $\mathbf{x}_{t-2}$, etc.

Consider that at each iteration of the denoising process, projected diffusion methods can be split into two steps:

1. **Gradient Step:** $\mathbf{x}'_t = \mathbf{x}_t + \gamma_t \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{s_t}$
2. **Projection Step:** $\mathbf{x}_{t-1} = \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t)$

These steps are sequentially applied in the reverse process to sample from a constrained subdistribution.

$$\mathbf{x}_t \rightarrow \underbrace{\mathbf{x}_t + \gamma_t s_t}_{\mathbf{x}'_t} \rightarrow \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t) = \mathbf{x}_{t-1} \rightarrow \underbrace{\mathbf{x}_{t-1} + \gamma_{t-1} s_{t-1}}_{\mathbf{x}'_{t-1}} \rightarrow \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_{t-1}) = \mathbf{x}_{t-2} \dots$$

By construction, $\mathbf{x}_{t-1} = \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t) \in \mathbf{C}$. Next, let us define the projection distance to $\mathbf{C}$ as:

$$f(\mathbf{x}) = D_{\text{KL}}(\mathbf{x}, \mathbf{C}) = D_{\text{KL}}(\mathbf{x} \| \mathcal{P}_{\mathbf{C}}(\mathbf{x}))$$

Since $\mathbf{C}$ is β -prox regular, by definition the following hold:

- f is differentiable outside $\mathbf{C}$ (in a neighborhood)
- $\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathcal{P}_{\mathbf{C}}(\mathbf{x}))$
- ∇f is L -Lipshitz with $L = \frac{2}{\beta}$

The standard “descent lemma” (or smoothness inequality) for L -smooth functions applies:

Lemma F.1. $\forall \mathbf{x}, \mathbf{y}$ in the neighborhood of $\mathbf{C}$:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2 = \boxed{f(\mathbf{x}) + 2\langle \mathbf{x} - \mathcal{P}_{\mathbf{C}}(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{\beta} \|\mathbf{y} - \mathbf{x}\|^2}$$

Applying this lemma, let us use $\mathbf{x} = \mathbf{x}'_{t-1}$ and $\mathbf{y} = \mathbf{x}'_t$. Noting that $\mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t) = \mathbf{x}_{t-1}$, we get:

$$D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) \leq \underbrace{D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C})}_{\text{Term A}} + \underbrace{2\langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \mathbf{x}'_t - \mathbf{x}'_{t-1} \rangle}_{\text{Term B}} + \underbrace{\frac{1}{\beta} \|\mathbf{x}'_t - \mathbf{x}'_{t-1}\|^2}_{\text{Term C}} \quad (\star)$$

Decomposing Term B. First, consider that since the step size is decreasing $\gamma_t \geq \gamma_{t-1}$:

$$\begin{aligned} \mathbf{x}'_{t-1} - \mathbf{x}_{t-2} &\leq (\mathbf{x}_{t-1} - \mathbf{x}_{t-2}) + \gamma_{t-1} s_{t-1} \\ &\leq (\mathbf{x}_{t-1} - \mathbf{x}_{t-2}) + \gamma_t s_{t-1} \end{aligned}$$

By the same rationale,

$$\mathbf{x}'_t - \mathbf{x}'_{t-1} \leq (\mathbf{x}_t - \mathbf{x}_{t-1}) + \gamma_t (s_t - s_{t-1}). \quad (\text{Definition B.1})$$

Proof of non-expansiveness of the projection operator. Next, we prove the non-expansiveness of the projection operator:

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| \leq 2\gamma_{t+1}G^2 \quad (\mathcal{L}^+)$$

Given $\mathbf{x}_t = \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_{t+1})$ and $\mathbf{x}_{t-1} = \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t)$,

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| = \|\mathcal{P}_{\mathbf{C}}(\mathbf{x}'_{t+1}) - \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_t)\| \leq \|\mathbf{x}_{t+1} - \mathbf{x}_t\|$$

since projections onto closed prox-regular sets are L -Lipshitz.

Now:

$$\begin{aligned} \mathbf{x}'_{t+1} &= \mathbf{x}_{t+1} + \gamma_{t+1} s_{t+1}; \\ \mathbf{x}'_t &= \mathbf{x}_t + \gamma_t s_t; \\ \mathbf{x}'_{t+1} - \mathbf{x}'_t &= (\mathbf{x}_{t+1} - \mathbf{x}_t) + (\gamma_{t+1} s_{t+1} - \gamma_t s_t). \end{aligned} \quad (\text{Definition B.2})$$

Making the projection residual,

$$\mathbf{x}_{t+1} - \mathbf{x}_t = \mathbf{x}_{t+1} - \mathcal{P}_{\mathbf{C}}(\mathbf{x}'_{t+1})$$

orthogonal to the target space at $\mathbf{x}_t$ (and any vector of the form $s_{t+1} - s_t$). Thus, since $\|s_t\| \leq G \ \forall t$:

$$\|\mathbf{x}'_{t+1} - \mathbf{x}'_t\|^2 = \|\gamma_{t+1} s_{t+1}\|^2 + \|\gamma_t s_t\|^2 \leq (\gamma_{t+1}^2 + \gamma_t^2) G^2$$

Taking the square root:

$$\|\mathbf{x}'_{t+1} - \mathbf{x}'_t\| \leq \sqrt{\gamma_{t+1}^2 + \gamma_t^2} G$$

Since $\gamma_{t+1} \geq \gamma_t$:

$$\begin{aligned} \|\mathbf{x}'_{t+1} - \mathbf{x}'_t\| &\leq \sqrt{2}\gamma_{t+1}G \\ &< 2\gamma_{t+1}G \end{aligned}$$

Finally, by applying Definition (B.1), $\|\mathbf{x}_{t+1} - \mathbf{x}_t\| \leq \|\mathbf{x}'_{t+1} - \mathbf{x}'_t\|$, and thus:

$$\boxed{\|\mathbf{x}_{t+1} - \mathbf{x}_t\| \leq 2\gamma_{t+1}G}$$

□

Now, prox-regularity gives:

$$\begin{aligned}\langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \mathbf{x}'_t - \mathbf{x}'_{t-1} \rangle &\leq \beta \|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2 \\ &\leq 4\beta\gamma_{t+1}^2 G^2\end{aligned}\quad (\text{Bound B.1})$$

where the Bound B.1 is derived by applying $(\mathcal{L}^+)$.

Since $\mathbf{C}$ in β -prox regular, for any point u near $\mathbf{C}$ and $v \in \mathbf{C}$:

$$\langle u - \mathcal{P}_{\mathbf{C}}(u), v - \mathcal{P}_{\mathbf{C}}(u) \rangle \leq \beta \|v - \mathcal{P}_{\mathbf{C}}(u)\|^2$$

Above, we substitute:

$$\begin{aligned}u &= \mathbf{x}'_t = \mathbf{x}_t + \gamma_t s_t \\ v &= \mathbf{x}_t \\ \mathcal{P}_{\mathbf{C}}(u) &= \mathbf{x}_{t-1}\end{aligned}$$

Now, expanding the inner product:

$$\begin{aligned}\langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \mathbf{x}'_t - \mathbf{x}'_{t-1} \rangle &= \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, (\mathbf{x}_t + \gamma_t s_t) - (\mathbf{x}_{t-1} + \gamma_{t-1} s_{t-1}) \rangle \\ &\leq \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, (\mathbf{x}_t - \mathbf{x}_{t-1}) + \gamma_t (s_t - s_{t-1}) \rangle \\ &\leq \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, (\mathbf{x}_t - \mathbf{x}_{t-1}) \rangle + \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \gamma_t (s_t - s_{t-1}) \rangle\end{aligned}$$

and since $\|s_t\| \leq G \quad \forall t: \langle s_{t+1}, s_t \rangle \leq \|s_{t+1}\| \|s_t\| \leq G^2$ so $\langle s_{t-1}, s_t \rangle - \|s_{t+1}\|^2 \leq G^2$, and:

$$\langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \gamma_t (s_t - s_{t-1}) \rangle \leq \gamma_t^2 G^2 \quad (\text{Bound B.2})$$

By applying Definition (B.2):

$$\begin{aligned}\langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \mathbf{x}'_t - \mathbf{x}'_{t-1} \rangle &= \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, (\mathbf{x}_t - \mathbf{x}_{t-1}) + (\gamma_t s_t - \gamma_{t-1} s_{t-1}) \rangle \\ &\leq \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, (\mathbf{x}_t - \mathbf{x}_{t-1}) \rangle\end{aligned}$$

Therefore, by applying Bound (B.1) to the previous inequality and Bound (B.2) directly, Term B is upper bounded by:

$$2 \langle \mathbf{x}'_{t-1} - \mathbf{x}_{t-2}, \mathbf{x}'_t - \mathbf{x}'_{t-1} \rangle \leq 8\beta\gamma_{t+1}^2 G^2 + 2\gamma_t^2 G^2 \quad (\text{Bound B.3})$$

Decomposing Term C. Next, we derive a bound on Term C in Eq. (*). As already shown,

$$\|\mathbf{x}'_t - \mathbf{x}'_{t-1}\| \leq 4\gamma_t G,$$

given:

$$\begin{aligned}\mathbf{x}'_t - \mathbf{x}'_{t-1} &\leq \underbrace{(\mathbf{x}_t - \mathbf{x}_{t-1})}_{\leq 2\gamma_{t+1} G} + \underbrace{\gamma_t (s_t - s_{t-1})}_{\leq 2G} \\ &\leq 4\gamma_{t+1} G\end{aligned}$$

Thus,

$$\frac{1}{\beta} \|\mathbf{x}'_t - \mathbf{x}'_{t-1}\|^2 \leq \frac{16}{\beta} \gamma_{t+1}^2 G^2 \quad (\text{Bound C.1})$$

Combining bounds (B.3) and (C.1) into (*), and recalling that $\gamma_{t+1} \geq \gamma_t$:

$$D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) \leq \underbrace{D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C})}_d + \underbrace{(8\beta + 2 + \frac{16}{\beta}) \gamma_{t+1}^2 G^2}_K$$

Now, we rewrite Term A, which for ease of notation we will refer to as d :

$$d = (1 - 2\beta\gamma_{t+1})d + 2\beta\gamma_{t+1}d$$

Thus:

$$\begin{aligned}D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) &\leq d - 2\beta\gamma_{t+1}d + 2\beta\gamma_{t+1}d + K\gamma_{t+1}^2 G^2 \\ &= (1 - 2\beta\gamma_{t+1})d + [2\beta\gamma_{t+1}d + K\gamma_{t+1}^2 G^2]\end{aligned}$$

Next, through Young's inequality, we simplify this expression further.

Theorem F.2. (Young's Inequality) $\forall u, v \geq 0, \epsilon > 0$:

$$uv \leq \frac{u^2}{2\epsilon} + \frac{\epsilon v^2}{2}$$

If we choose $u = \sqrt{2\beta\gamma_{t+1}}d$, $v = \sqrt{K}\gamma_{t+1}G$, and $\epsilon = \frac{2\beta d}{K\gamma_{t+1}G^2}$, then

$$\begin{aligned} uv &= \sqrt{2\beta\gamma_{t+1}}d \times \sqrt{K}\gamma_{t+1}G \\ &= \sqrt{2K}\gamma_{t+1}^{\frac{3}{4}}Gd \end{aligned}$$

Applying Young's Inequality:

$$\begin{aligned} uv &\leq \frac{u^2}{2\epsilon} + \frac{\epsilon v^2}{2} \\ &= \frac{2\beta\gamma_{t+1}d}{2(\frac{2\beta d}{K\gamma_{t+1}G^2})} + \frac{\epsilon v^2}{2} \\ &= \frac{K\gamma_{t+1}^2G^2}{2} + \frac{\epsilon v^2}{2} \\ &= \frac{K\gamma_{t+1}^2G^2}{2} + \left(\frac{1}{2} \times \frac{2\beta d}{K\gamma_{t+1}G^2} \times K\gamma_{t+1}^2G^2\right) \\ &= \frac{K\gamma_{t+1}^2G^2}{2} + \beta\gamma_{t+1}d \end{aligned}$$

Thus,

$$\sqrt{2K}\gamma_{t+1}^{\frac{3}{4}}Gd \leq \frac{K\gamma_{t+1}^2G^2}{2} + \beta\gamma_{t+1}d$$

Finally, taken altogether:

$$\begin{aligned} 2\beta\gamma_{t+1}d + K\gamma_{t+1}^2G^2 &\leq \beta\gamma_{t+1}d + \left(\frac{K\gamma_{t+1}^2G^2}{2} + \beta\gamma_{t+1}d\right) \\ &= 2\beta\gamma_{t+1}d + \frac{K}{2}\gamma_{t+1}^2G^2 \end{aligned}$$

Since $\gamma_{t+1} \leq \frac{\beta}{2G^2}$, then

$$\frac{K}{2}\gamma_{t+1}^2G^2 \leq \frac{1}{2} \left(8\beta + 2 + \frac{16}{\beta}\right) \frac{\beta^2}{4G^2} = \mathcal{O}(\beta^3)$$

which is bounded by $\gamma_{t+1}^2G^2$ for all $\beta \geq 0$.

Thus,

$$2\beta\gamma_{t+1}d + K\gamma_{t+1}^2G^2 \leq 2\beta\gamma_{t+1}d + \gamma_{t+1}^2G^2.$$

By substitution we obtain:

$$D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) \leq (1 - 2\beta\gamma_{t+1})D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C}) + \underbrace{\gamma_{t+1}^2G^2}_{\mathcal{O}(\beta^3)}$$

Finally, we reparameterize this, such that $\alpha_t = 2\beta\gamma_t$, implying $\gamma_t = \frac{\alpha_t}{2\beta}$. Plugging this in, we get:

$$\begin{aligned} D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) &\leq (1 - \alpha_t)D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C}) + \gamma_{t+1}^2G^2 \\ &\leq (1 - \alpha_t)D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C}) + \alpha_{t+1}^2G^2 \end{aligned}$$

and thus,

$$\boxed{D_{\text{KL}}(\mathbf{x}'_t, \mathbf{C}) \leq (1 - \alpha_t)D_{\text{KL}}(\mathbf{x}'_{t-1}, \mathbf{C}) + \alpha_{t+1}^2G^2}$$

□