GENERATION-FREE VERIFICATION IN DIFFUSION LANGUAGE MODELS VIA ANCHOR CHAIN SCANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

As language models solve increasingly complex reasoning problems, the need for reliable verification grows accordingly. Notably, a single error can propagate through a reasoning trajectory and yield a plausible yet incorrect answer. To address this challenge, recent work has explored generative verifiers based on autoregressive language models and diffusion language models (DLMs). However, such generative methods can hallucinate during verification and often waste computation on unnecessary parts of the reasoning trajectory. To overcome this, we propose DE-ANCHORING: a generation-free DLM-based verifier that focuses on information anchor chains, defined as repeated occurrences of key information throughout a reasoning trajectory. DE-ANCHORING simultaneously masks all occurrences of each anchor and uses their DLM reconstruction probabilities to measure how strongly the remaining reasoning supports the original information. Across practical verification tasks, including Best-of-N selection, unsupervised error diagnosis, and auxiliary reward signals for reinforcement learning, DE-ANCHORING substantially outperforms generative verification baselines. Moreover, DE-ANCHORING remains robust under aggressive cost reduction, achieving comparable performance with only 12.5% of the generative verification cost.

1 INTRODUCTION

Recent autoregressive language models (ARMs) have demonstrated remarkable performance across a wide range of tasks (Wei et al., 2023; Kojima et al., 2023; Wang et al., 2023). However, they continue to suffer from hallucination, which poses a serious concern in high-stakes domains where factually accurate decision-making is critical (Ji et al., 2023; Lin et al., 2022). A key factor in hallucination is the stochastic, left-to-right decoding of ARMs: an initially hallucinated token can bias subsequent generations to justify it, thereby accumulating the error through "*hallucination snowballing*" (Zhang et al., 2023). Reasoning tasks are particularly susceptible to this limitation because intermediate outputs are repeatedly referenced in subsequent steps, causing early errors to accumulate and distort the final answer. Accordingly, a range of verification methods have been proposed to address this issue, including evaluation by external models (Zheng et al., 2023; Liu et al., 2023), repeated sampling (Wang et al., 2023), and self-critique generation (Madaan et al., 2023). However, these verification methods developed for ARMs require generating new outputs, which incurs high computational costs and the risk of introducing new hallucinations.

Meanwhile, diffusion language models (DLMs) have recently gained increasing attention, as DLMs can understand bidirectional context and generate multiple tokens simultaneously at arbitrary positions (Nie et al., 2025; Ye et al., 2025). Beyond its efficiency, this structure naturally supports self-verification through flexible regeneration (Zhong et al., 2026; Ruan et al., 2026). In particular, Ruan et al. (2026) leverage DLMs for self-verification, showing that their intrinsic reconstruction dynamics provide an effective verification signal for reasoning tasks. However, this approach remains computationally expensive, similar to generative methods for ARMs; it requires repeated multi-step trajectory denoising and multiple full regenerations to reduce stochastic variance. More importantly, as a generative method, it can also introduce new hallucinations during the generation. Consequently, this leads us to a simple question: **does verification really require generation?**

Contribution. We propose a new generation-free verification framework for DLMs, DE-ANCHORING. Our framework builds on two key insights. First, the flexible masking mechanism

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

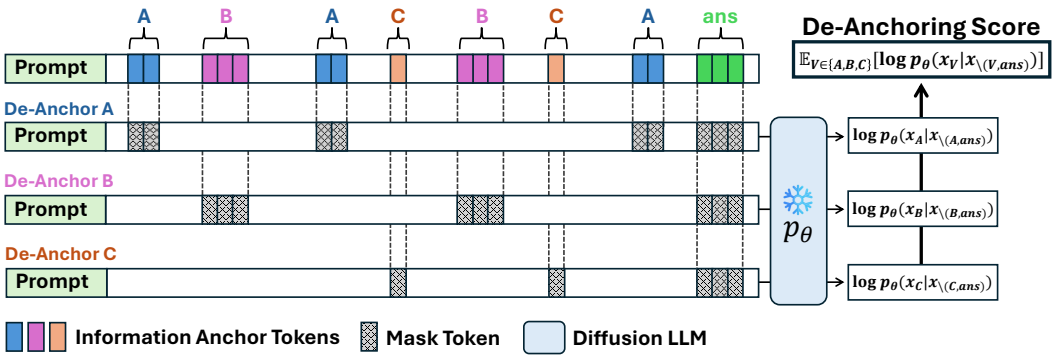


Figure 1: Overview of DE-ANCHORING. Given a reasoning trace, $V \in \{A, B, C\}$ denotes an information anchor. For each anchor, all of its occurrences in the trajectory (called a *reference chain*) are masked simultaneously together with the answer span (*ans*), followed by one forward pass of the frozen DLM p_θ. The anchor-level score $\log p_\theta(x_V \mid x_{\setminus(V,\text{ans})})$ (Eq. 4) asks how well the model recovers the anchor's original tokens at every position of its reference chain at once, with no remaining copy of the anchor to lean on. The final trace-level score is the average of the anchor-level scores over all anchors A, B, and C; higher indicates more consistent reasoning.

of DLMs allows verification to be reframed as identifying and probing only the information necessary to assess a generated sequence, rather than regenerating the entire sequence. Second, reliable reasoning should make its key *information anchors* (*e.g.*, derived values in mathematical reasoning) logically recoverable from the surrounding derivation, rather than relying on superficial repetition.

Building on this insight, DE-ANCHORING quantifies how reliably an information anchor can be recovered across its occurrences when they are simultaneously masked. Specifically, given a reasoning trace, DE-ANCHORING constructs a *structurally corrupted* context for each information anchor by simultaneously masking all occurrences of that anchor. This removes direct cues from repeated mentions, allowing us to probe whether the remaining reasoning supports the anchor. In a single forward pass per anchor, DE-ANCHORING reads out the log-probabilities of the original tokens at the masked positions, measuring their recoverability without explicitly generating replacements. We aggregate these log-probabilities across anchors into a trace-level verification score.

Across two popular DLMs, LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025), we evaluate DE-ANCHORING on nine reasoning benchmarks across three downstream verification settings: Best-of-N selection (Cobbe et al., 2021; Lightman et al., 2023), unsupervised error diagnosis (Ruan et al., 2026; Bradley, 1997; Davis & Goadrich, 2006), and reinforcement learning (Shao et al., 2024; Zhao et al., 2025). In Best-of-N selection, DE-ANCHORING improves accuracy over the non-verification Semi-AR baseline by up to $+19.5\%$p. For unsupervised error diagnosis, it achieves AUROC of up to 0.819 and AUPR of up to 0.939, respectively. When used as an auxiliary reward for reinforcement learning, DE-ANCHORING further improves upon standard reinforcement learning, yielding gains of up to $+3.0\%$p over *diffu*-GRPO (Zhao et al., 2025) alone. These results show that DLM-based simultaneous verification that targets the information anchor chain enables effective reasoning verification without full regeneration.

2 RELATED WORK

Diffusion Language Models. Diffusion models have recently extended their impact on natural language generation, following their success in image generation (Ho et al., 2020; Song et al., 2022). Conventional ARMs generate text sequentially from left to right. In contrast, DLMs generate text through an iterative denoising process, allowing them to generate an arbitrary number of tokens at arbitrary positions at each step. Early work on DLMs mainly explored text generation with continuous diffusion (Austin et al., 2023; Li et al., 2022; Dieleman et al., 2022). However, these approaches were difficult to scale effectively. Masked-diffusion-based DLMs later emerged as a more promising alternative, and they have recently achieved performance comparable to autoregressive models of similar scale (Nie et al., 2025; Ye et al., 2025).

DLM-based Verification. Recent studies have further explored the potential of DLMs for generation-based correction and verification (Wang et al., 2025; Dong et al., 2025; Lee et al., 2025; Zhong et al., 2026; Ruan et al., 2026). *ReMDM* (Wang et al., 2025) and Saber (Dong et al., 2025) enable iterative correction during generation based on model confidence. *HEX* (Lee et al., 2025) exploits diverse generation orders and verifies outputs through agreement across generations under different block schedules. *DiSE* (Zhong et al., 2026) performs verification using token regeneration probabilities over the entire generated sequence, given the full context. *BMC* (Ruan et al., 2026) measures reasoning correctness through stability in fully reconstructing randomly masked context.

3 PRELIMINARIES

Recent studies (Nie et al., 2025; Shi et al., 2025; Arriola et al., 2025; Seo et al., 2025) use a common probabilistic formulation for diffusion language models (DLMs). Let $t \in (0, 1]$ denote a continuous time variable that controls the degree of corruption in the forward process. The forward process gradually corrupts the original sequence x_0 by independently replacing each token with a special mask token M with probability t. This produces a partially masked sequence x_t, with more tokens becoming masked as t increases. At $t = 1$, the sequence is fully masked. The reverse process reconstructs the sequence in the opposite direction. It starts from the fully masked sequence at $t = 1$ and gradually recovers the original tokens as t approaches 0.

To learn this reverse process, the model is trained on partially masked sequences. Given x_t, the model predicts the original token at every masked position in a single forward pass. Specifically, if a token in x_t has been replaced by M, the model $p_\theta(\cdot \mid x_t)$ is trained to predict the corresponding token in the clean sequence x_0. The training loss is therefore computed only at the masked positions:

$$\mathcal{L}(\theta) \triangleq - \mathop{\mathbb{E}}_{t, x_0, x_t} \left[\frac{1}{t} \sum_{i=1}^{L} \mathbb{1}\left[x_t^i = \mathrm{M}\right] \log p_\theta\left(x_0^i \mid x_t\right) \right], \tag{1}$$

where x_0 is a training sequence of length L, $t \sim \mathcal{U}(0, 1]$ denotes the masking ratio, and x_t is sampled from the corresponding forward corruption process. The indicator $\mathbb{1}[\cdot]$ selects the positions that are masked in x_t, while the factor $1/t$ normalizes the loss by the expected fraction of masked tokens. Importantly, ground-truth sequences in the training data are generally expected to be logically consistent. As a result, sequences containing **internal contradictions** are less typical under the learned training distribution and behave differently during inference. We build our method on this insight.

4 METHOD

Overview. In this section, we propose DE-ANCHORING, an efficient DLM-based validity-scoring method that selectively verifies the core information flow of reasoning. DE-ANCHORING builds on the hypothesis that verifying repeated tokens carrying key information, which we call *information anchors*, can provide a useful signal for assessing reasoning quality.

For example, in mathematical reasoning, derived values can directly affect subsequent reasoning and the final answer, whereas discourse markers such as "Therefore" can often be replaced with "So" or "Thus" without changing the meaning or final judgment. Accordingly, the verification process should focus *selectively* on the parts of the reasoning that can meaningfully affect the final outcome. This selective focus also enables efficient verification by avoiding the need to regenerate the entire response. However, such targets are often scattered across the reasoning trace and appear multiple times. DLMs are particularly well suited to this setting, as they allow arbitrary subsets of tokens to be masked and predicted within the full reasoning context. Building on this insight, we first categorize the types of tokens that can serve as information anchors for different reasoning tasks.

4.1 MOTIVATING EXPERIMENT

For a systematic and comprehensive categorization of tokens, we classify them across nine reasoning benchmarks. These benchmarks cover a broad range of reasoning tasks and domains, including mathematical reasoning, multiple-choice reasoning across diverse subject areas, and open-ended code generation. Specifically, we define token categories including task-specific tokens (a *derived value*, an *option letter*, or an *identifier*), semantic content tokens (content words and material tokens

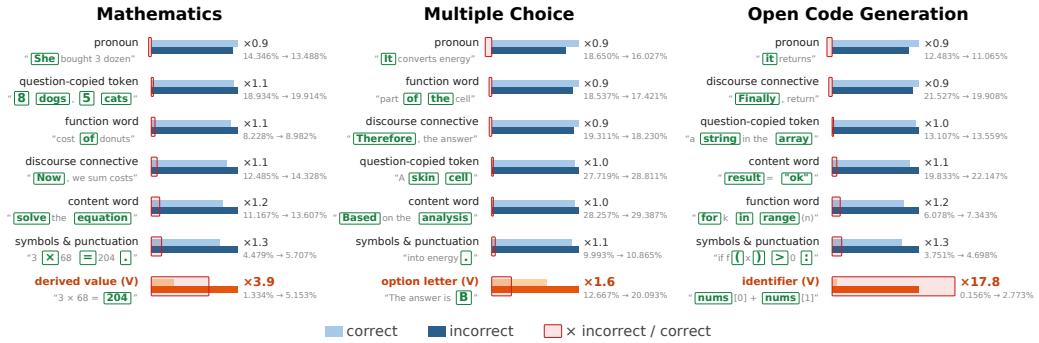


Figure 2: Rejection signals over token classes. The token classes cover 99.72–99.86% of all token occurrences. For each token class, we mask all occurrences of each token in the reasoning and perform a single forward pass, visualizing the mean probability mass assigned to alternative tokens (i.e., all tokens except the original token), which reflects the model's tendency to reject the original token. Relative to correct reasoning, this rejection tendency increases substantially in incorrect reasoning for specific classes: *derived value* in Mathematics (3.9×), *option letter* in Multiple Choice Reasoning (1.6×), and *identifier* in Open Code Generation (17.8×).

copied from the question), and fluency tokens (function words, discourse connectives, pronouns, and symbols). These token categories achieve 99.72–99.86% coverage across all datasets (Table 8). Rather than relying on a probabilistic tagging model, we use a deterministic, dictionary-style categorization based on the set of words occurring across the entire dataset, and provide the corresponding rule ordering in Table 10. When evaluated post-hoc using an NLTK POS tagger (Bird & Loper, 2004), the resulting categorizations show a high level of agreement (Table 9). This provides empirical support for the validity of our deterministic categorization scheme.

Next, under the hypothesis that verifying specific types of tokens carrying key information can provide useful verification signals, we assess each token class separately for correct and incorrect reasoning. Specifically, for each token class, we simultaneously mask all occurrences of the same token in the reasoning trajectory to control for repetition effects and isolate the verification signal. We then measure the probability mass assigned to alternative tokens other than the original token; this probability reflects the model's tendency to reject the original generation.

The results in Figure 2 show that task-specific anchor classes (V) exhibit the largest increase in preference for rejecting the original token in incorrect reasoning relative to correct reasoning. This suggests that rejection signals in the reasoning trajectory are concentrated in these classes. Interestingly, these classes are indeed closely associated with reasoning quality. For example, in mathematical tasks such as GSM8K (Cobbe et al., 2021), the most important information arguably appears in the form of derived values (*e.g.*, the value of C in $A \times B = C$), since errors in these values (*e.g.*, arithmetic mistakes) can immediately degrade the reasoning quality. Accordingly, we treat tokens in these classes as *information anchors* that provide key information to assess reasoning trajectory.

Surprisingly, the rejection signal of an *information anchor chain*, a structured set of repeated anchors, reflects a systematic understanding of the reasoning error. As shown in Figure 3, jointly masking the anchor chain of the erroneous token 384 shifts preference to the correct value, 204. This preference is supported by higher attention mass to both the earlier operands (3, 68) and the later error-derived result 874. Anchors near the operands strongly reject 384, while those near 874 retain more probability on it, indicating internal contradiction. This suggests that simultaneous verification can quantify inconsistency within an anchor chain.

Overall, these observations indicate that DLMs can quantify the internal consistency of a reasoning trajectory through flexible mask context construction, enabling a verification procedure that scans repeated anchor occurrences in a single forward pass.

4.2 De-Anchoring: Verification via Scanning Information Anchor Chains

Building on these observations, we now formalize De-Anchoring, which translates the identified consistency signal into a verification score over information anchor chains (Figure 1 for overview).

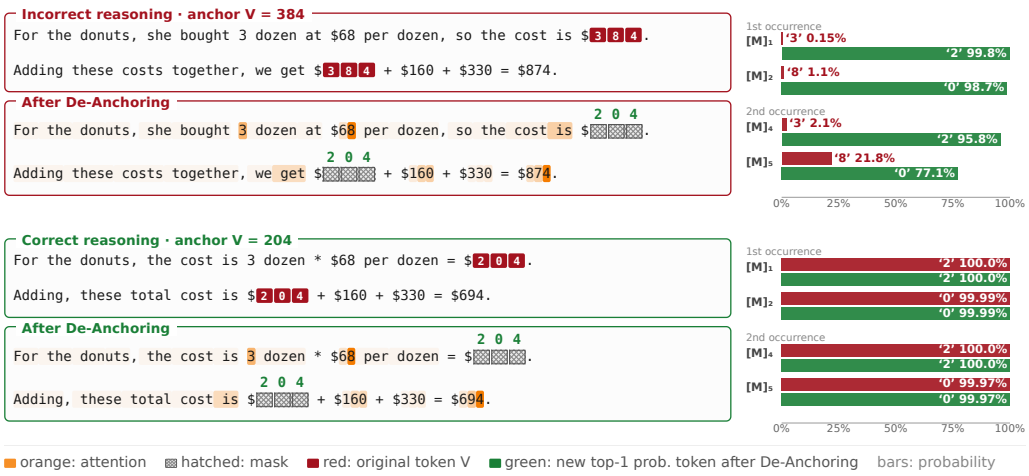


Figure 3: DE-ANCHORING sharply lowers the original-token probability in the anchor chain when the reasoning is inconsistent, as shown in example #11 from the GSM8K test set. While consistent reasoning maintains 99.97–100%, inconsistent reasoning drops as low as 0.15%, demonstrating that DE-ANCHORING strongly tracks reasoning validity. For a long anchor chain example, see Figure 5.

Given a pre-generated reasoning trajectory x with answer span ans, a task-specific parser D extracts m different information anchors from it, ordered by first occurrence and truncated to at most K:

$$D(x) = \{a_i\}_{i=1}^{m}, \qquad m \le K. \tag{2}$$

For each anchor a_i, let $r_1, \ldots, r_{n_i}$ denote the contiguous token spans corresponding to its repeated n_i occurrences in x, such that $x[r_j] = a_i$ for each j, where $x[r_j]$ denotes the subsequence of x indexed by the span r_j. The corresponding anchor chain V_i and the collection over anchors V^* are

$$V_i = \bigcup_{j=1}^{n_i} r_j, \qquad V^* = \{V_i\}_{i=1}^{m}. \tag{3}$$

Then, we construct $x_{\setminus(V_i,\mathrm{ans})}$ by masking every token of each anchor chain V_i at once, together with the answer span ans. After that, for each chain V_i, we measure the recovery of the original anchor tokens by averaging the log-probabilities of all masked tokens x_l in the chain:

$$\log p_\theta\big(x_{V_i} \mid x_{\setminus(V_i,\mathrm{ans})}\big) := \frac{1}{|V_i|} \sum_{l \in V_i} \log p_\theta\big(x_l \mid x_{\setminus(V_i,\mathrm{ans})}\big)\,. \tag{4}$$

The final verification score is obtained by averaging the recovery scores across all anchor chains:

$$S(x) = \mathbb{E}_{V_i \in V^*}\left[\log p_\theta\big(x_{V_i} \mid x_{\setminus(V_i,\mathrm{ans})}\big)\right]. \tag{5}$$

Each chain requires a single forward pass through the frozen DLM p_θ. Thus, scoring one candidate requires m forward passes, where $m \le K$.

Beyond contradiction detection through simultaneous masking (Figure 3), masking the full reference chain of an informative anchor provides another benefit: preventing self-copying. If not all repeated occurrences are masked, the model could simply copy an existing error from the remaining chain, leading to an overestimation of its preference for the original token. As shown in Table 4, the gap between masking the full chain and a partial chain grows to as much as 5%p, highlighting the importance of full-chain masking for reliable verification. The answer span is also always masked during verification; when the answer is left visible, verifying the reasoning trajectory can introduce noise by allowing intermediate computations to be adjusted toward the fixed final outcome, a behavior also discussed by Lee et al. (2025). Consistent with this, Table 14 shows that answer span masking is generally beneficial, with modest gains.

In summary, DE-ANCHORING is a reasoning verifier that scans **structurally corrupted anchor chains**. It scores logical self-consistency in a post hoc manner; a low DE-ANCHORING score indicates that the remaining bidirectional context fails to recover the reference chain consistently, suggesting internal inconsistency. In contrast, correct reasoning tends to preserve logical self-consistency and thus receives a higher score.

Table 1: Best-of-N selection performance comparison of verifiers on `LLaDA-8B-Instruct` and `Dream-7B-Instruct`. Results are reported as accuracies (%). **Bold** and underline indicate the best and second-best results, respectively. See Table 11 for temperature-based sampling results.

Method	GSM8K	MATH	Countdown	ARC-C	TruthfulQA	BBH	MMLU	HumanEval	MBPP
LLaDA-8B-Instruct									
Semi-AR	78.39	30.20	17.58	78.92	52.19	37.60	56.00	34.76	41.63
Valid-Format Random	78.52	32.01	17.39	83.05	53.08	40.21	58.83	36.97	45.71
HEX (Self-Consistency)	82.79	34.40	23.05	84.81	54.09	44.00	59.00	39.02	46.69
Self-Evaluation	78.47	29.80	17.97	78.92	50.88	37.20	57.00	35.37	44.36
Self-Confidence	78.39	33.00	26.95	80.29	54.97	40.20	56.40	33.54	40.08
BMC	82.39	34.07	25.39	84.64	54.24	47.47	60.53	38.82	46.95
De-Anchoring	**84.38**	**34.80**	**29.69**	**85.49**	**56.58**	**48.80**	**60.80**	**40.24**	**51.75**
Dream-7B-Instruct									
Semi-AR	78.24	40.60	19.92	83.11	54.82	50.40	67.00	35.98	51.36
Valid-Format Random	78.30	38.81	28.74	85.75	57.41	50.20	69.12	45.99	56.32
HEX (Self-Consistency)	84.23	43.20	25.39	87.12	59.50	**53.40**	70.20	43.29	57.98
Self-Evaluation	76.72	39.20	22.66	82.94	54.24	49.00	68.00	36.59	47.86
Self-Confidence	76.35	36.80	32.81	82.85	54.68	50.20	67.60	35.98	50.19
BMC	83.93	42.40	34.38	87.51	58.28	52.73	69.80	46.34	58.63
De-Anchoring	**85.75**	**43.40**	**39.45**	**87.63**	**61.99**	53.00	**71.80**	**51.22**	**61.09**

5 Experiments

5.1 Setups

Models, Sampling, and Datasets. We adopt `LLaDA-8B-Instruct` (Nie et al., 2025) and `Dream-7B-Instruct` (Ye et al., 2025) as base models, using standard semi-autoregressive (semi-AR) sampling (Nie et al., 2025) with an output length of 256 tokens. To construct diverse and robust reasoning traces from these models, we use block-based sampling (Lee et al., 2025) with block lengths of 2, 4, 8, 16, and 32 at temperature 0, and Gumbel-noise sampling (Zheng et al., 2025) with a block length of 32 and temperature 0.6 across five random seeds. We unmask **two** tokens per forward step, requiring 128 steps per sequence in all settings and thereby ensuring equal generation cost. All experiments are conducted on a single NVIDIA B200 GPU. We evaluate on nine reasoning benchmarks spanning three categories: (1) *mathematics* (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), and Countdown), (2) *multiple-choice reasoning* (ARC-Challenge (Clark et al., 2018), TruthfulQA (Lin et al., 2022), BBH (Suzgun et al., 2022), and MMLU (Hendrycks et al., 2021a)), and (3) *open code generation* (HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021)), with the code benchmarks evaluated by executing unit tests.

Metrics and Baseline Verifiers. For each input query, we generate $N{=}5$ candidate responses and report Best-of-N accuracy based on the candidate selected by the verifier. Following (Ruan et al., 2026), we additionally report AUROC (Bradley, 1997) and AUPR (Davis & Goadrich, 2006), treating correct responses as the positive class, to measure the verifier's ability to distinguish correct from incorrect responses. For ablation studies, we consider **only** queries whose candidate sets contain both correct and incorrect responses. Following prior work (Ruan et al., 2026), we compare against two ARM-based verification methods applied to DLMs: *Self-Confidence* (Kadavath et al., 2022) and *Self-Evaluation* (Madaan et al., 2023). We also include *HEX* (Lee et al., 2025), a block-based Self-Consistency (Wang et al., 2023) verifier for DLMs, and *BMC* (Ruan et al., 2026), a generative DLM verifier. To avoid distorted evaluation from candidates that cannot be parsed because of formatting or instruction-following failures, we also include a random selector that samples uniformly from candidates with valid formatting in the generation pool (*Valid-Format Random*).

5.2 Main Result

As shown in Tables 1 and 2, De-Anchoring consistently achieves strong performance across nine reasoning benchmarks on both `LLaDA-8B-Instruct` and `Dream-7B-Instruct`, outperforming prior ARM and DLM-based verifier baselines in most settings. All results use response pools constructed with block-based sampling (see Appendix B for the results with temperature-

Table 2: Unsupervised Error Diagnosis Performance on `LLaDA-8B-Instruct` and `Dream-7B-Instruct`. Each entry reports AUROC / AUPR. The response pool includes candidates generated with block lengths $\{2, 4, 8, 16, 32\}$ for each question. **Bold** and underline indicate the best and second-best results, respectively. See Table 12 for temperature-based sampling results.

Method	GSM8K	MATH	Countdown	ARC-C	TruthfulQA	BBH	MMLU	HumanEval	MBPP
LLaDA-8B-Instruct									
Self-Evaluation	0.505 / 0.792	0.498 / 0.358	0.571 / 0.217	0.538 / 0.846	0.441 / 0.506	0.506 / 0.421	0.514 / 0.603	0.517 / 0.358	0.632 / 0.517
Self-Confidence	0.561 / 0.819	0.637 / 0.511	0.600 / 0.268	0.583 / 0.865	0.515 / 0.546	0.545 / 0.474	0.554 / 0.627	0.646 / 0.455	0.621 / 0.534
BMC	0.741 / 0.889	0.759 / 0.652	**0.662** / 0.297	0.661 / 0.893	0.542 / 0.546	0.562 / 0.460	0.599 / 0.668	0.660 / 0.422	0.690 / 0.569
De-Anchoring	**0.798 / 0.928**	**0.761 / 0.675**	0.638 / **0.469**	**0.729 / 0.917**	**0.664 / 0.671**	**0.666 / 0.536**	**0.659 / 0.719**	**0.739 / 0.467**	**0.815 / 0.699**
Dream-7B-Instruct									
Self-Evaluation	0.518 / 0.798	0.527 / 0.466	0.555 / 0.350	0.528 / 0.866	0.514 / 0.582	0.501 / 0.533	0.512 / 0.701	0.554 / 0.412	0.485 / 0.522
Self-Confidence	0.594 / 0.840	0.594 / 0.537	0.635 / 0.470	0.551 / 0.871	0.528 / 0.583	0.586 / **0.620**	0.526 / 0.705	0.617 / 0.490	0.660 / 0.658
BMC	0.728 / 0.889	0.723 / 0.662	0.621 / 0.422	0.641 / 0.897	0.550 / 0.603	0.579 / 0.587	0.565 / 0.731	0.656 / 0.517	0.672 / 0.667
De-Anchoring	**0.782 / 0.918**	**0.806 / 0.768**	**0.704 / 0.645**	**0.751 / 0.939**	**0.659 / 0.700**	**0.589** / 0.611	**0.659 / 0.801**	**0.819 / 0.672**	**0.807 / 0.795**

based sampling). Notably, we conduct a budget-matched comparison with *BMC*, a state-of-the-art generative verification method for DLMs.

We find that De-Anchoring is especially effective for long and complex reasoning tasks. Longer chains involve more self-reference as illustrated in Figure 5, making inconsistencies more likely to be penalized. As a result, Best-of-N selection is more likely to retain internally consistent reasoning trajectories. For example, Countdown shows large gains over the verification-free Semi-AR baseline: $+12.11\%$p for `LLaDA-8B-Instruct` and $+19.53\%$p for `Dream-7B-Instruct` (Table 1), where these gains drop sharply under partial corruption of the anchor chain (Section 6.1). This highlights the importance of evaluating the entire long chain at once for detecting subtle errors.

DE-ANCHORING also outperforms *HEX*, a Self-Consistency voting method, by up to $+14.06\%$p on Countdown. Furthermore, on open-ended code generation tasks, DE-ANCHORING outperforms *HEX* by up to $+7.93\%$p and $+5.06\%$p on HumanEval and MBPP, respectively. This suggests that DE-ANCHORING mitigates a key failure mode of majority voting: selecting plausible but incorrect majority answers. Unlike Self-Consistency, DE-ANCHORING can preserve a small number of correct answers by assigning them much higher scores than the erroneous majority.

This is also consistent with the strong AUPR in Table 2: DE-ANCHORING tends to assign higher scores to correct reasoning than to contradictory trajectories. This is evident on Unsupervised Error Diagnosis, where DE-ANCHORING exceeds state-of-the-art DLM-based verifier *BMC* by up to $+0.125$ AUROC and $+0.172$ AUPR on `LLaDA-8B-Instruct`, and by up to $+0.163$ AUROC and $+0.223$ AUPR on `Dream-7B-Instruct`. These results suggest that DE-ANCHORING enables reliable reasoning selection and opens the door to using it as an auxiliary reward signal for reinforcement learning (RL).

6 ABLATION STUDY AND ANALYSIS

DE-ANCHORING verifies reasoning trajectories by examining informative anchor chains. While precisely targeting the core information *within* each chain is crucial for reliable verification, computational cost can be reduced by limiting verification *across* chains. Section 6.1 shows that inaccurate targeting substantially degrades DE-ANCHORING performance, whereas Section 6.2 shows that performance can be largely preserved using only a small number of complete anchor chains. Lastly, Section 6.3 shows that the DE-ANCHORING score can serve not only as a test-time verifier but also as an auxiliary RL reward, allowing its signal to be internalized into the model parameters.

6.1 VERIFICATION SCOPE WITHIN AN ANCHOR CHAIN

Verify Only Informative Anchors. To examine whether information anchor class provides key signals for verification, we conduct an anchor replacement ablation. Table 3 shows that applying DE-ANCHORING outside the informative anchor class V leads to a marked drop in verification performance. Rows pool the datasets of each family (*mathematics*: GSM8K, MATH, Countdown; *multiple choice reasoning*: ARC-C, TruthfulQA, BBH, MMLU; *open code generation*: HumanEval, MBPP). Verification performance decreases by up to 24.11%p in BoN, 0.409 in AUROC, and 0.203 in AUPR when the anchor is replaced by a class that carries a less informative signal of the reasoning.

Table 3: Anchor-class replacement results. Each entry reports BoN / AUROC / AUPR, pooled over the datasets of the corresponding family. Each anchor class is defined in Table 9. Candidates are responses decoded at block sizes $b \in \{2, 4, 8, 16, 32\}$.

family	V			QCopy			Sym.			Content			Function			Conn.			Pron.		
LLaDA-8B-Instruct																					
Mathematics	**70.72**	**0.721**	**0.791**	62.16	0.585	0.727	53.30	0.516	0.678	53.00	0.510	0.674	52.85	0.504	0.675	53.75	0.511	0.684	52.85	0.545	0.740
Multiple Choice	**67.12**	**0.611**	**0.729**	58.88	0.517	0.681	60.26	0.531	0.677	57.41	0.497	0.669	60.45	0.522	0.680	55.05	0.469	0.662	57.70	0.494	0.689
Open Code Generation	**82.98**	**0.887**	**0.894**	70.21	0.549	0.735	72.34	0.562	0.752	64.54	0.493	0.715	72.34	0.560	0.748	60.28	0.478	0.691	58.87	0.556	0.788
Dream-7B-Instruct																					
Mathematics	**79.86**	**0.766**	**0.846**	67.77	0.552	0.747	61.37	0.520	0.726	64.10	0.500	0.722	65.17	0.526	0.729	60.31	0.476	0.703	60.19	0.524	0.742
Multiple Choice	**77.18**	**0.620**	**0.751**	73.04	0.510	0.704	73.49	0.518	0.700	71.91	0.516	0.697	72.21	0.511	0.695	72.29	0.506	0.703	71.99	0.497	0.701
Open Code Generation	**84.62**	**0.846**	**0.879**	71.28	0.483	0.726	80.51	0.606	0.785	70.77	0.470	0.723	73.85	0.498	0.727	66.67	0.617	0.788	68.21	0.517	0.747

Table 4: Full-Chain robustness on mathematical reasoning datasets. j denotes the maximum number of copies left unmasked, with j=0 corresponding to DE-ANCHORING. Each cell pools GSM8K, MATH and Countdown. For $j \geq 1$, up to j unmasked copies are chosen at random, averaged over three seeds.

	LLaDA-8B-Instruct					**Dream-7B-Instruct**				
Metric	j=0	j=1	j=2	j=3	j=4	j=0	j=1	j=2	j=3	j=4
BoN	70.7%	67.7%	67.2%	66.5%	65.9%	79.9%	77.1%	75.8%	75.1%	74.9%
AUROC	0.721	0.697	0.690	0.690	0.685	0.766	0.732	0.717	0.709	0.705
AUPR	0.791	0.770	0.765	0.764	0.761	0.846	0.827	0.819	0.814	0.813

Verify the Full Anchor Chain. Table 4 shows that masking all repeated occurrences of an anchor is important for reliable DE-ANCHORING. Relative to $j = 0$ (zero copies left unmasked, *i.e.*, setup of DE-ANCHORING), verification performance generally degrades as j increases (*i.e.*, more of the informative chain is left visible). This is because visible contradictory anchors can provide false support for the same contradiction at the verified position, making the original token appear more plausible. This creates cross-reference among repeated copies rather than an independent verification signal. The verification can thus overestimate contradictory tokens in this case, confirming the need for full-chain verification in DE-ANCHORING with all repeated anchors masked.

Verify Anchors and Nothing Else. To isolate the effect of the clean information anchor chain, we add redundant random masking to the verification context. Table 5 shows that DE-ANCHORING consistently loses performance when the verification context is further corrupted by masking an additional 1–128 random positions. Values are pooled across the three mathematics datasets for LLaDA and Dream; rows with additional random masking report performance averaged over three random masking patterns. This degradation generally becomes more pronounced as the amount of random masking increases, causing the key information carried by the anchor chain to occupy a smaller fraction of the overall corruption and become increasingly diluted by noise.

Table 5: Effect of additional random masking on mathematical reasoning datasets (GSM8K, MATH, Countdown).

	BoN	**AUROC**	**AUPR**
Anchor Chain	**75.8%**	**0.745**	**0.820**
+ 1 random	75.7%	0.744	0.818
+ 2 random	75.7%	0.743	0.819
+ 4 random	75.7%	0.744	0.818
+ 8 random	75.4%	0.741	0.816
+ 16 random	75.0%	0.737	0.814
+ 32 random	75.4%	0.736	0.815
+ 64 random	74.5%	0.727	0.808
+ 128 random	72.8%	0.711	0.795

6.2 DE-ANCHORING'S ROBUSTNESS UNDER REDUCED COMPUTATIONAL COST

To evaluate how aggressively the computational cost of verification can be reduced, we vary the number of verified chains over $K \in \{8, 4, 2, 1\}$, following Eqs. 2–5, and perform forward passes only on the selected chains. This setting is well matched to the structure of the reasoning traces: even in mathematics and open-ended code generation, where cross-chain referencing is most frequent, the average number of informative anchor chains is around five. Also, only a small fraction of examples contain more than eight chains (Figure 4). Table 6 shows that DE-ANCHORING remains robust across these increasingly selective verification settings and under different chain-selection strategies (1st–6th rows). Remarkably, even at the most aggressive setting of K=1, DE-ANCHORING slightly outperforms *BMC* despite requiring substantially fewer forward passes.

Table 7: Performance comparison of `LLaDA-8B-Instruct` with the same diffu-GRPO training configuration. Ext. Model denotes the use of an external embedding model (Ruan et al., 2026).

Method	Ext. Model	GSM8K	MATH	Countdown
`LLaDA-8B-Instruct`		78.24	31.60	18.36
+ ***diffu*-GRPO**	–	78.77 (+0.53)	32.40 (+0.80)	23.83 (+5.47)
+ ***diffu*-GRPO + BMC**	✓	79.61 (+1.37)	31.80 (+0.20)	22.27 (+3.91)
+ ***diffu*-GRPO + DE-ANCHORING**	–	**81.73** (+3.49)	**33.00** (+1.40)	**24.61** (+6.25)

We attribute this robustness at small K to the fact that not every anchor chain needs to be directly verified to influence the verification score. When DE-ANCHORING verifies a selected target anchor chain, only the occurrences of that anchor are masked, while all other anchor chains remain visible in the conditioning context. Therefore, if an unselected chain is invalid, the resulting inconsistency in the overall reasoning context can still affect the reconstruction probabilities of the masked target through bidirectional attention of the DLM. As a result, a contradictory chain can lower the DE-ANCHORING score of a directly verified chain even when the contradictory chain itself is not selected for verification. Thus, while preventing self-copying *within* a chain remains necessary for obtaining a meaningful verification signal (Section 6.1), verifying every chain is not. This allows DE-ANCHORING to reduce verification cost from at most 6.25% ($K = 8$) to as little as 0.78% ($K = 1$) of the per-candidate generation cost in the pool, with little loss in effectiveness.

Table 6: Lightweight verification. Micro-averaged Best-of-N performance across all model–dataset settings for varying numbers of anchor chains K.

Method	K=1	K=2	K=4	K=8
First K	66.71	67.20	67.66	67.86
Last K	66.95	67.37	67.84	67.90
Random K	66.62	67.17	67.71	67.90
Most Frequent K	67.26	67.36	67.84	**67.93**
Least Frequent K	66.30	66.74	67.35	67.80
Top K by Total Tokens	**67.30**	**67.70**	**67.89**	67.90
Semi-AR	61.26	61.26	61.26	61.26
Self-Evaluation	60.94	60.94	60.94	60.94
Self-Confidence	61.87	61.87	61.87	61.87
Valid-Format Random	63.20	63.20	63.20	63.20
HEX (Self-Consistency)	65.81	65.81	65.81	65.81
BMC (N_{BMC}=1)	65.54	65.79	65.97	66.03
BMC (N_{BMC}=2)	–	65.91	66.10	66.13
BMC (N_{BMC}=4)	–	–	65.93	66.10
BMC (N_{BMC}=8)	–	–	–	66.06

Together with this robustness to lightweight verification, the strong AUPR in Table 2 suggests that DE-ANCHORING can reduce verification cost while still effectively prioritizing high-quality reasoning trajectories. This reliable high-score regime also makes the verification score a natural auxiliary reward signal for reinforcement learning (RL), as validated in the next section.

6.3 DE-ANCHORING AS AN AUXILIARY REWARD FOR REINFORCEMENT LEARNING

Our preceding analyses show that the DE-ANCHORING score strongly penalizes contradictory reasoning while reliably identifying high-quality trajectories, as reflected by its strong AUPR. This suggests that it can serve not only as a test-time verifier but also as an auxiliary RL reward capturing reasoning signals complementary to standard objectives. We provide details of the training configuration in Appendix C. Table 7 shows that augmenting *diffu*-GRPO (Zhao et al., 2025) with rewards for trajectories ranked highly by DE-ANCHORING improves performance beyond *BMC*, a DLM-based generative verification signal. Moreover, unlike *BMC*, which requires an external model (Reimers & Gurevych, 2019), DE-ANCHORING achieves larger gains under a more constrained computational budget. This result demonstrates that fine-grained reasoning signals of DE-ANCHORING can be efficiently and effectively learned through RL.

7 CONCLUSION

We propose DE-ANCHORING, an efficient DLM-based verifier that focuses computation on the anchor chains carrying the core information flow. We show that DLMs' verification capability can be activated by selectively masking information anchor chains identified through simple rule-based classification, without costly external probabilistic models. Across nine reasoning benchmarks, DE-ANCHORING outperforms existing generative verifiers in most settings for both Best-of-N selection and error diagnosis. Furthermore, it provides fine-grained verification signals that complement conventional reinforcement learning objectives. Together, these results establish DE-ANCHORING as an efficient and robust framework for reasoning verification.

AI USE STATEMENT

We used generative AI tools for translation, writing assistance, code development, experiment execution, and data analysis. All AI-assisted outputs were reviewed by the authors, who made the final decisions. We take full responsibility for this work, including all AI-assisted content and artifacts.

ETHICS STATEMENT

Regarding the societal impact of this research, our work introduces a cost-efficient approach for verifying AI-generated reasoning by activating the latent verification capabilities of Diffusion Language Models. Compared with generative verifiers, DE-ANCHORING substantially reduces the computation required for verification. By lowering inference overhead, this approach may also reduce the associated energy consumption and environmental footprint of large-scale reasoning verification.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we transparently provide the theoretical motivation and implementation details of our method in Sections 3, 4, and 5. For additional implementation details required for reproduction but not covered in the main text, refer to the Appendix. We will release our code upon acceptance.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models, 2025. URL https://arxiv.org/abs/2503.09573.

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

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.

Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pp. 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/P04-3031/.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(96)00142-2. URL https://www.sciencedirect.com/science/article/pii/S0031320396001422.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 233–240, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143874. URL https://doi.org/10.1145/1143844.1143874.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, Fei Huang, Yongbin Li, and Ge Li. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model, 2025. URL https://arxiv.org/abs/2510.18165.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, March 2023. ISSN 1557-7341. doi: 10.1145/3571730. URL http://dx.doi.org/10.1145/3571730.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Jihoon Lee, Hoyeon Moon, Kevin Zhai, Arun Kumar Chithanar, Anit Kumar Sahu, Soummya Kar, Chul Lee, Souradip Chakraborty, and Amrit Singh Bedi. Test-time scaling in diffusion llms via hidden semi-autoregressive experts, 2025. URL https://arxiv.org/abs/2510.05040.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL https://arxiv.org/abs/2303.16634.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL https://arxiv.org/abs/1908.10084.

Jiaoyang Ruan, Xin Gao, Yinda Chen, Hengyu Zeng, Liang Du, Guanghao Li, Jie Fu, and Jian Pu. Reasoning on the manifold: Bidirectional consistency for self-verification in diffusion language models, 2026. URL https://arxiv.org/abs/2604.16565.

Yeongbin Seo, Dongha Lee, Jaehyung Kim, and Jinyoung Yeo. Fast and fluent diffusion language models via convolutional decoding and rejective fine-tuning, 2025. URL https://arxiv.org/abs/2509.15188.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling, 2025. URL https://arxiv.org/abs/2503.00307.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

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

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball, 2023. URL https://arxiv.org/abs/2305.13534.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.12216.

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, 2025. URL https://arxiv.org/abs/2409.02908.

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.

Linhao Zhong, Linyu Wu, Wen Wang, Yuling Xi, Chenchen Jing, Jiaheng Zhang, Hao Chen, and Chunhua Shen. Efficient self-evaluation for diffusion language models via sequence regeneration, 2026. URL https://arxiv.org/abs/2603.02760.

APPENDIX

A Information Anchor Classification Details **15**

B De-Anchoring Performance under Temperature-Based Sampling **16**

C De-Anchoring as an Auxiliary Reward for Reinforcement Learning **17**

D Anchor Chain Count Distribution **17**

E Answer-Span Masking Ablation **18**

F Qualitative Examples **19**

A Information Anchor Classification Details

Table 8: Coverage of all token categories. The unclassified 0.14–0.28% is excluded; it includes broken tokens, such as single-character BPE fragments or split multi-byte characters (`U+FFFD`).

Dataset	Covered by all categories	Unclassified
Mathematics (GSM8K, MATH, Countdown)	**99.72%**	0.28%
Multiple Choice (ARC-C, TruthfulQA, BBH, MMLU)	**99.82%**	0.18%
Open Code Generation (HumanEval, MBPP)	**99.86%**	0.14%

Table 9: Token categories. Green reports **post-hoc agreement** with an NLTK POS tagger on the four categories where part of speech applies. * counts sentence-initial *Next* as a connective, as it is the only word the tagger disagrees on there. The 4-letter floor is a simple heuristic for open-class membership: without it, *the*, *of*, *a*, *is* occur in essentially every question and would absorb up to 57% of all function-word occurrences into the question-copied class.

category	Mathematics	Multiple-Choice	Open Code Generation
V — derived value	a computed number absent from the question (48 in *16 × 3 = 48*)	—	—
V — option letter	—	an option letter A–E in the response (article *a*/*A* excluded)	—
V — identifier	—	—	an identifier where its value is used (AST `Name` in Load context)
question-copied	a number, or an open-class word of 4+ letters, already present in the question	a word of 4+ letters present in the question	a word of 4+ letters present in the question
symbols & punctuation	operators between digits, =, punctuation, LaTeX braces, markup	punctuation, LaTeX braces, `<answer>` and ``` markup	punctuation and markup, plus operators and brackets inside the code block
content	open-class words; unit words count here — *dollars, dollar, cents, hours, hour, minutes, minute, days, day, weeks, week, months, year, years, miles, pounds, ounces, cups, inches, feet, meters, percent* (22) NLTK 99.8%	open-class words, response numbers included, option letters excluded NLTK 99.5%	open-class prose, identifiers with _ or a digit, numeric and string literals NLTK 99.5%
function	closed-class words (96) — determiners, prepositions, conjunctions, auxiliaries, quantifiers: *the, a, an, of, to, in, is, are, was, were*, … NLTK 97.4%	closed-class words (96) — determiners, prepositions, conjunctions, auxiliaries, quantifiers: *the, a, an, of, to, in, is, are, was, were*, … NLTK 98.7%	the same 96 closed-class words, plus Python builtins, and keywords inside the code block NLTK 98.4%
connective	*Therefore, Thus, Hence, So, Consequently, Accordingly, Then, Next, First, Finally, However, Also, Now, Since, Because* (15); infinitive *to* excluded NLTK 100.0%*	*Therefore, Thus, Hence, So, Consequently, Accordingly, Then, Next, First, Finally, However, Also, Now, Since, Because* (15); infinitive *to* excluded NLTK 100.0%*	*Therefore, Thus, Hence, So, Consequently, Accordingly, Then, Next, First, Finally, However, Also, Now, Since, Because* (15); infinitive *to* excluded NLTK 100.0%
pronoun	*He, She, They, It, he, she, they, it, his, her, their, him, them, its* (14 entries, matched case-insensitively) NLTK 100.0%	*He, She, They, It, he, she, they, it, his, her, their, him, them, its* (14 entries, matched case-insensitively) NLTK 100.0%	*He, She, They, It, he, she, they, it, his, her, their, him, them, its* (14 entries, matched case-insensitively) NLTK 100.0%

Table 10: Rule order, first match wins. Keywords and string literals apply only inside the code block, so a prose *if* or *return* stays an ordinary word. For Mathematics, question-copied is not a rule: copied numbers, like V, come directly from parsing, while copied words are relabeled from the content and unit rules afterwards.

family	rule order
Mathematics (GSM8K, MATH, Countdown)	**V** (derived value) → symbols (operators between digits) → symbols (=) → content (unit words) → pronoun → connective → function → content (other words) → symbols (punctuation, markup)
Multiple Choice (ARC-C, TruthfulQA, BBH, MMLU)	**V** (option letter) → question-copied → content (numbers) → pronoun → connective → function → content (other words) → symbols (punctuation, markup)
Open Code Generation (HumanEval, MBPP)	**V** (identifier, AST `Name`/Load) → function (Python builtins) → content (number literals) → content (string literals, in code) → function / symbols (keywords / operators, in code) → question-copied → content (prose words) → function → pronoun → connective → symbols (punctuation, markup) → content (remaining names)

B De-Anchoring Performance under Temperature-Based Sampling

Table 11: Best-of-N selection performance comparison of verifiers on `LLaDA-8B-Instruct` and `Dream-7B-Instruct`. Results are reported as accuracies (%). **Bold** and underline indicate the best and second-best results, respectively. For the corresponding results on the block-based sampling pool, see Table 1.

Method	GSM8K	MATH	Countdown	ARC-C	TruthfulQA	BBH	MMLU	HumanEval	MBPP
LLaDA-8B-Instruct									
Semi-AR	79.30	31.80	14.84	78.41	51.46	42.60	53.20	34.76	36.96
Valid-Format Random	78.39	31.00	18.36	82.59	53.22	43.40	59.60	35.37	40.47
HEX (Self-Consistency)	84.46	34.40	19.92	85.07	55.85	45.60	**60.20**	38.41	43.19
Self-Evaluation	79.61	31.40	18.75	78.92	49.12	42.00	52.40	34.15	41.63
Self-Confidence	78.32	33.20	25.78	76.88	46.64	43.60	55.80	35.98	38.52
BMC	83.52	33.00	27.08	84.78	54.39	43.33	59.47	36.18	45.65
De-Anchoring	**86.28**	**34.60**	**30.86**	**85.41**	**58.19**	**52.20**	60.00	**43.29**	**50.97**
Dream-7B-Instruct									
Semi-AR	75.51	37.40	23.83	82.42	54.39	49.80	65.80	37.20	44.75
Valid-Format Random	77.63	34.80	28.12	86.18	55.26	48.80	71.20	47.56	50.58
HEX (Self-Consistency)	87.11	42.20	33.20	88.14	**60.67**	**53.60**	74.00	49.39	54.47
Self-Evaluation	76.88	35.60	31.64	83.79	53.80	50.40	69.40	38.41	46.69
Self-Confidence	75.36	34.00	31.64	82.08	54.82	48.40	67.40	44.51	49.81
BMC	84.46	40.73	36.20	86.72	57.46	52.20	70.53	47.97	53.70
De-Anchoring	**87.19**	**43.40**	**42.58**	**88.57**	60.53	52.80	**74.40**	**53.05**	**63.42**

Table 12: Unsupervised Error Diagnosis Performance on `LLaDA-8B-Instruct` and `Dream-7B-Instruct`. Each entry reports AUROC / AUPR. The response pool combines five candidates generated for each question. **Bold** and underline indicate the best and second-best results, respectively. For the corresponding results on the block-based sampling pool, see Table 2.

Method	GSM8K	MATH	Countdown	ARC-C	TruthfulQA	BBH	MMLU	HumanEval	MBPP
LLaDA-8B-Instruct									
Self-Evaluation	0.506 / 0.791	0.496 / 0.344	0.545 / 0.198	0.535 / 0.840	0.447 / 0.515	0.524 / 0.461	0.524 / 0.602	0.566 / 0.395	0.637 / 0.494
Self-Confidence	0.576 / 0.829	0.643 / 0.506	0.610 / 0.277	0.585 / 0.862	0.508 / 0.541	0.538 / 0.491	0.574 / 0.639	0.614 / 0.449	0.605 / 0.481
BMC	0.738 / 0.893	0.763 / 0.614	0.663 / 0.299	0.629 / 0.877	0.560 / 0.575	0.536 / 0.468	0.585 / 0.654	0.669 / 0.467	0.640 / 0.511
De-Anchoring	**0.805 / 0.928**	**0.789 / 0.688**	**0.694 / 0.523**	**0.722 / 0.911**	**0.672 / 0.689**	**0.674 / 0.580**	**0.669 / 0.718**	**0.741 / 0.487**	**0.780 / 0.631**
Dream-7B-Instruct									
Self-Evaluation	0.518 / 0.784	0.518 / 0.405	0.598 / 0.330	0.527 / 0.861	0.501 / 0.566	0.511 / 0.514	0.507 / 0.704	0.505 / 0.400	0.476 / 0.474
Self-Confidence	0.596 / 0.824	0.600 / 0.494	0.633 / 0.393	0.568 / 0.881	0.491 / 0.551	0.588 / **0.606**	0.541 / 0.728	0.622 / 0.491	0.622 / 0.590
BMC	0.746 / 0.890	0.714 / 0.601	0.673 / 0.445	0.615 / 0.888	0.560 / 0.598	0.572 / 0.569	0.610 / 0.766	0.640 / 0.536	0.649 / 0.633
De-Anchoring	**0.791 / 0.919**	**0.785 / 0.711**	**0.680 / 0.555**	**0.759 / 0.939**	**0.682 / 0.708**	**0.615** / 0.596	**0.674 / 0.812**	**0.814 / 0.701**	**0.846 / 0.828**

C De-Anchoring as an Auxiliary Reward for Reinforcement Learning

	BMC	**De-Anchoring (ours)**
Reward $r_{\text{aux}}(c)$	$\mathbf{1}[c \text{ correct}] \cdot \alpha_t M \cdot \text{clip}(S(c), 0, 1)$	$\alpha_t M$ for the top $\lfloor k/2 \rfloor$ of the $k \geq 2$ correct rollouts of a group, ranked by $S(c)$
Score $S(c)$	Ruan et al. (2026)	Section 4.2
Budget per rollout	8 forward passes	≤ 8 forward passes
α_t	$0.5 \rightarrow 1.0$	1.0
External model	`all-MiniLM-L6-v2`	None

Table 13: Two auxiliary rewards added to diffu-GRPO: *BMC* and De-Anchoring. As reflected in the number of forward passes and *BMC*'s use of the external model `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019), De-Anchoring operates under a more constrained computational budget.

Notation. A rollout c is one sampled completion, and a group is the G=6 rollouts drawn for the same prompt, over which *diffu*-GRPO normalizes the advantage. $\mathbf{1}[\cdot]$ is the indicator, M=1 the reward scale, and α_t the curriculum coefficient at training step t out of T, annealed linearly for *BMC* and held constant for ours. $S(c)$ is the score of the corresponding verifier, and k denotes the number of correct rollouts in a group for which the score is defined. Both rewards are 0 for an incorrect rollout and for one whose score is undefined; ours is additionally 0 for a correct rollout outside the top $\lfloor k/2 \rfloor$ and for the whole group when $k < 2$. Both use the same frozen `LLaDA-8B-Instruct` scorer, kept separate from the policy; *BMC* additionally uses `all-MiniLM-L6-v2`.

RL setup. Each benchmark is trained on its own in-domain training split. Every run shares the same budget of $T = 4{,}000$ optimization steps at group size $G = 6$ with $\mu = 12$ inner iterations per generation round; each round draws 3 prompts (18 sequences at $G = 6$), covering about 1,000 distinct prompts. The policy is `LLaDA-8B-Instruct` with a rank-128 LoRA adapter over a 4-bit base, trained at learning rate 3×10^{-6} under a KL coefficient of 0.04 to the frozen base model as the reference policy, and generates at most 256 tokens with 128 diffusion steps at block length 32. The three trained arms of Table 7 share this configuration and differ only in r_{aux}.

D Anchor Chain Count Distribution

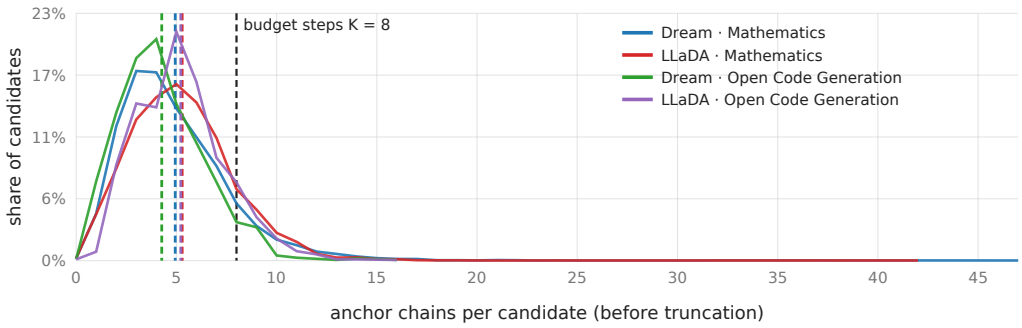


Figure 4: Number of informative anchor chains per reasoning trace. Most counts cluster around a similar range, with only a few long-tail cases. Accordingly, we use $K = 8$ by default, i.e., at most eight forward passes.

E ANSWER-SPAN MASKING ABLATION

Sample sizes. n counts items that contain both correct and incorrect candidates and is used as the denominator for Best-of-N selection (BoN). n_{AUROC} is smaller because a candidate enters AUROC / AUPR only when DE-ANCHORING yields a score for it, and an item enters only when both a correct and an incorrect candidate remain. A candidate is dropped when the answer span cannot be parsed, which affects both settings. In *multiple-choice reasoning* tasks (ARC-C, TruthfulQA, BBH, and MMLU), it is also dropped when the anchor lies entirely within the answer span, so that leaving the answer visible leaves no occurrence to mask. Both settings are compared using the same set of evaluable candidates, ensuring that Δ reflects the effect of answer visibility rather than differences in the candidate pool.

Answer span length effect. Answer span masking is more effective when the answer is short, as in GSM8K and *multiple-choice reasoning* tasks (ARC-C, TruthfulQA, BBH, and MMLU). Countdown is a special case in which the output span is substantially longer than in the other datasets, averaging 9.1 tokens. For *open-ended code generation* tasks, we treat the answer span as undefined because the nature of the task makes it difficult to identify a clear final answer span.

Table 14: Answer-span masking ablation. Each row averages `LLaDA-8B-Instruct` and `Dream-7B-Instruct`, weighted by n for BoN and n_{AUROC} for AUROC / AUPR; deltas are computed from unrounded values. *Masked* is the default policy, which masks the answer span together with every occurrence of the anchor chain; *Visible* leaves the answer span in place.

	Block pool ($b = 2, 4, 8, 16, 32$)										
	Judged items		BoN (%)			AUROC			AUPR		
Dataset	n	n_{AUROC}	Masked	Visible	Δ	Masked	Visible	Δ	Masked	Visible	Δ
GSM8K	907	787	82.69	81.48	-1.21	0.781	0.770	-0.011	0.861	0.855	-0.006
MATH	339	261	65.19	65.49	+0.29	0.698	0.704	+0.006	0.767	0.771	+0.003
Countdown	264	238	65.91	67.80	+1.89	0.677	0.698	+0.021	0.744	0.765	+0.021
ARC-C	783	144	72.54	71.01	-1.53	0.656	0.573	-0.083	0.803	0.756	-0.047
TruthfulQA	601	236	67.72	67.55	-0.17	0.627	0.611	-0.016	0.753	0.751	-0.002
BBH	538	80	55.95	55.95	+0.00	0.557	0.559	+0.002	0.746	0.756	+0.010
MMLU	425	114	67.06	63.53	-3.53	0.585	0.496	-0.088	0.745	0.683	-0.062
	Temperature pool ($\tau = 0.6$, 5 seeds)										
	Judged items		BoN (%)			AUROC			AUPR		
Dataset	n	n_{AUROC}	Masked	Visible	Δ	Masked	Visible	Δ	Masked	Visible	Δ
GSM8K	1,103	1,044	84.68	83.77	-0.91	0.795	0.790	-0.006	0.876	0.873	-0.003
MATH	369	316	63.69	62.87	-0.81	0.684	0.679	-0.005	0.760	0.757	-0.003
Countdown	301	297	61.79	64.12	+2.33	0.664	0.676	+0.012	0.716	0.729	+0.013
ARC-C	925	188	72.86	70.81	-2.05	0.665	0.600	-0.066	0.808	0.773	-0.035
TruthfulQA	732	285	60.66	56.01	-4.64	0.602	0.498	-0.104	0.757	0.700	-0.057
BBH	580	79	54.83	54.14	-0.69	0.584	0.496	-0.089	0.750	0.721	-0.029
MMLU	460	117	60.22	57.83	-2.39	0.553	0.484	-0.068	0.722	0.686	-0.036

F QUALITATIVE EXAMPLES

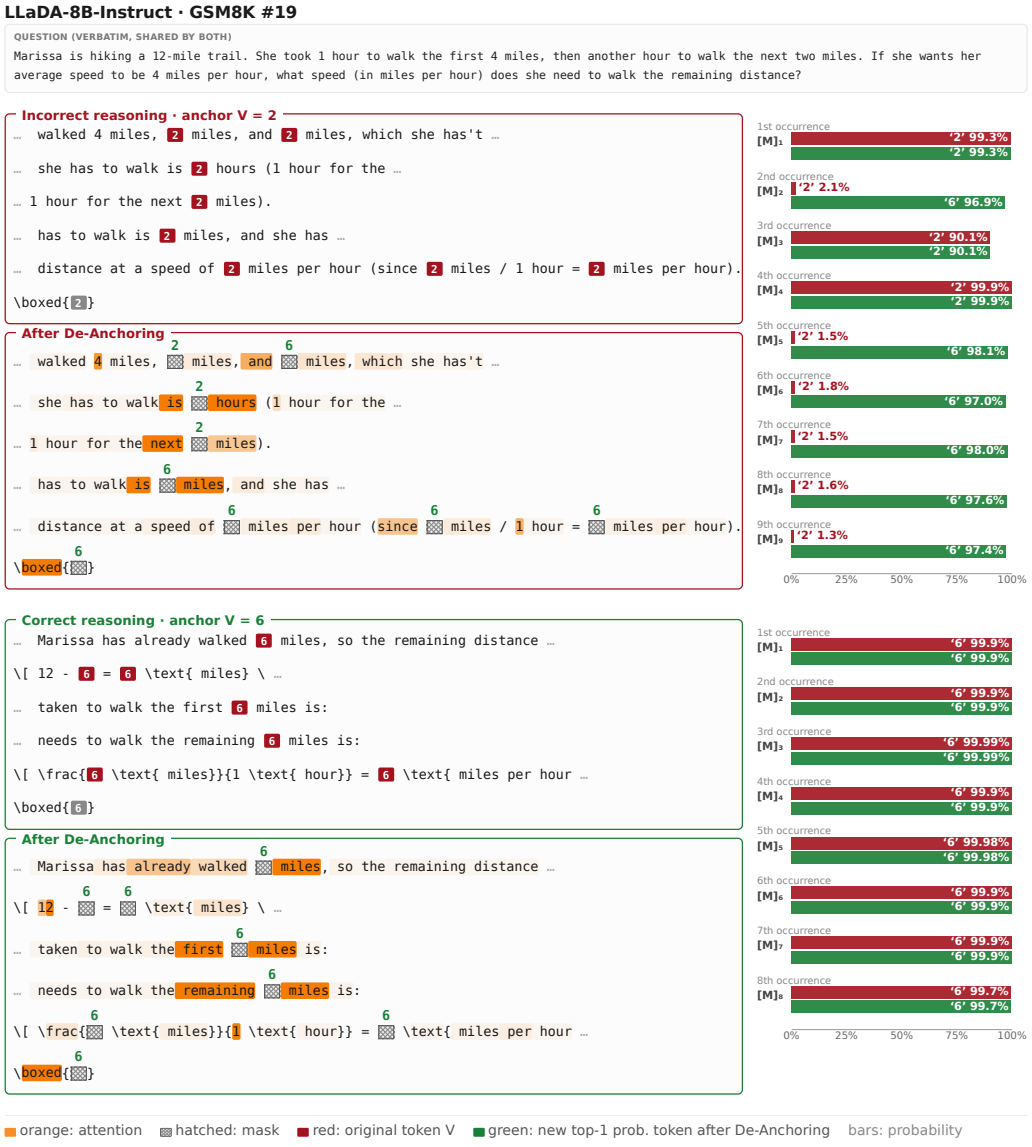


Figure 5: De-Anchoring on example #19 from the GSM8K test set with `LLaDA-8B-Instruct`, where the anchor chain is long. The incorrect reasoning chain (red) repeats the anchor value $V = 2$ at nine positions; after De-Anchoring the original-digit probability falls to 1.3–2.1% at six of them and the model recovers "6" instead, while three keep "2". The correct reasoning chain (green) repeats $V = 6$ at eight positions and holds between 99.68% and 100.0% at every position, with no top-1 prediction changing.

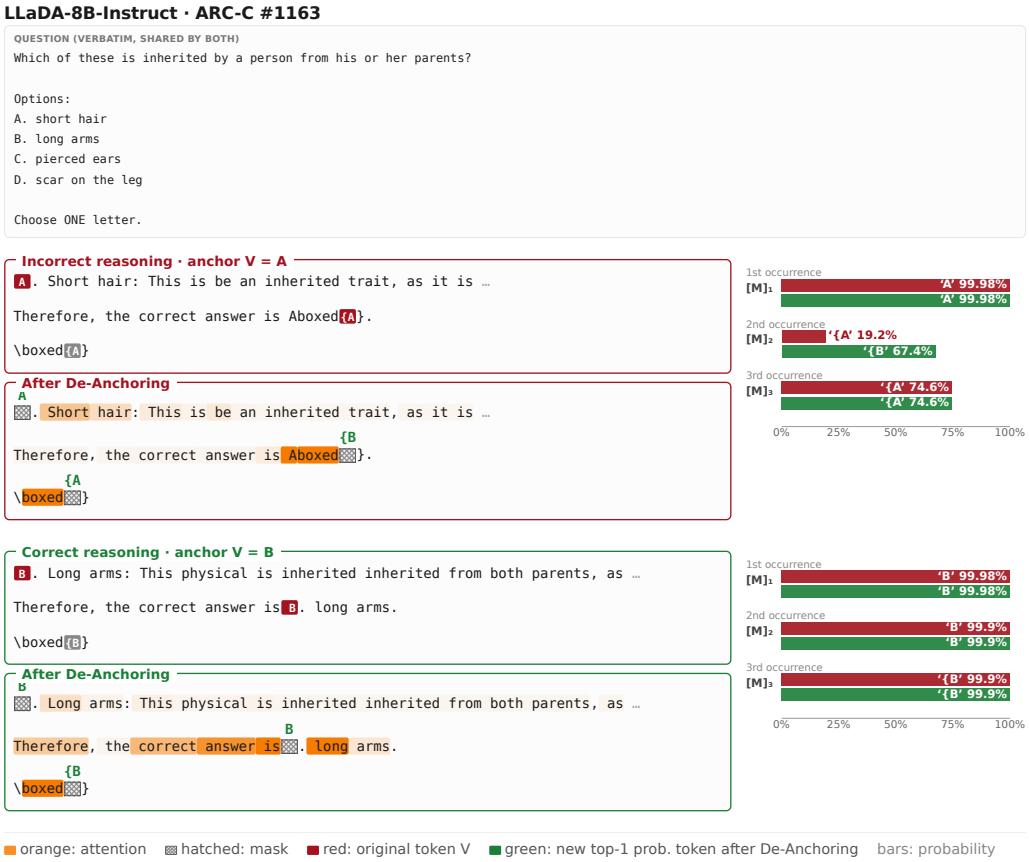


Figure 6: De-Anchoring on example #1163 from the ARC-Challenge test set with `LLaDA-8B-Instruct`, where the anchor is the chosen option letter and one of its occurrences is the answer span itself. The incorrect reasoning chain (red) commits to $V = \text{A}$, but at the `\boxed{` position the probability of "A" is only 19.20% and the top-1 prediction becomes "B" at 67.38%, while the first and third occurrences of "A" remain unchanged. The correct reasoning chain (green) commits to $V = \text{B}$, with the original-token probability remaining between 99.91% and 99.98% at all three positions.

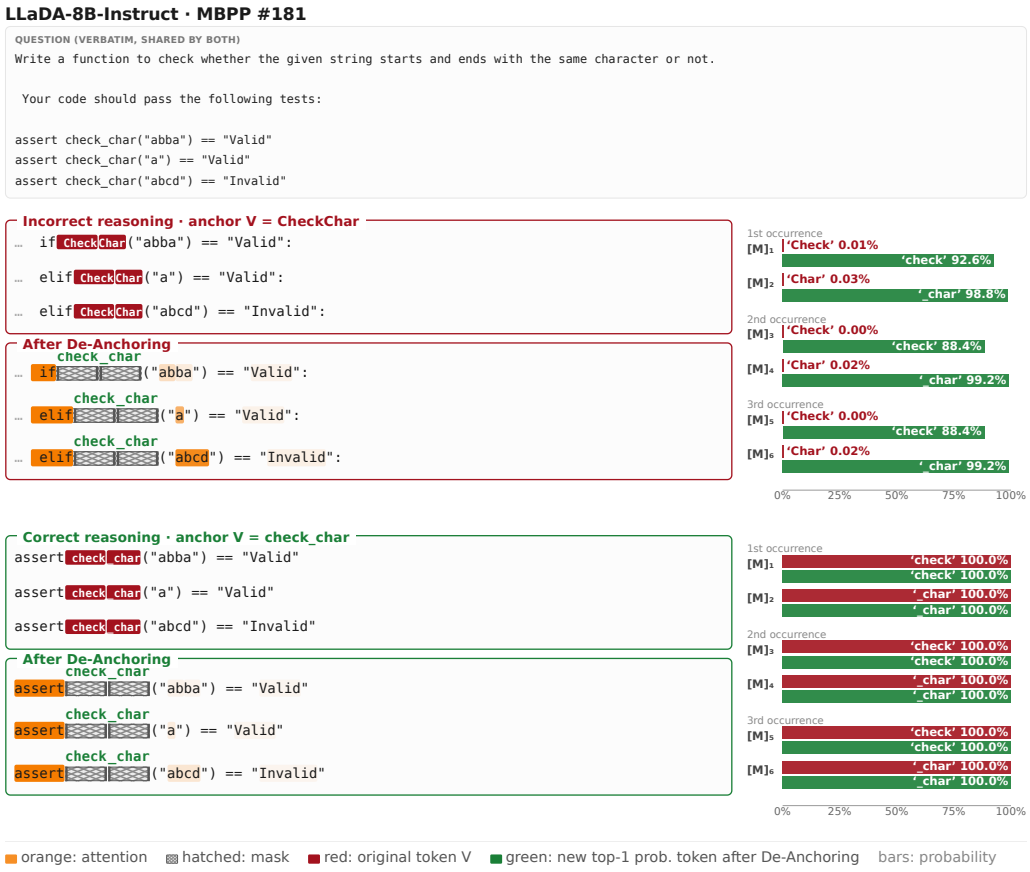


Figure 7: De-Anchoring on example #181 from the MBPP test set with `LLaDA-8B-Instruct`. Open-ended code generation has no answer span, so only the anchor chain is masked. The incorrect reasoning chain (red) invents the identifier $V =$ `CheckChar` instead of the `check_char` required by the given tests; after De-Anchoring the probability of the original subwords collapses to between 0.00% and 0.03% at all six masked positions and every one of the three occurrences is recovered as `check_char`. The correct reasoning chain (green) uses $V =$ `check_char`, with the original-token probability staying at 100% at all six positions and no top-1 prediction changing.