GENERATION ORDER AND PARALLEL DECODING IN MASKED DIFFUSION MODELS: AN INFORMATION-THEORETIC PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion models (MDMs) can accelerate sequence generation by decoding multiple positions in parallel, but two distributional questions remain central: how generation order interacts with model error, and what sampling bias is introduced by parallel factorization. We develop an information-theoretic framework separating *order sensitivity* from *parallelization bias*. Our analysis yields three insights. First, local conditional errors are evaluated under model-induced rollouts, so early mistakes can alter the contexts of later predictions. Second, factorized parallel decoding can produce individually likely but jointly implausible samples even with exact marginal predictions; forward KL, reverse KL, and incoherence reveal different aspects of this error. Third, exact correction by fixed whole-block rejection has expected proposal cost at least exponential in conditional total correlation. We study Prefix-Consistent Parallel Decoding (PCPD) as a practical relaxation that checks local prefix consistency rather than exact joint sampling. A controlled Block-HMM isolates sampling error, LLaDA provides ordering and parallelization diagnostics, and batched WeDLM experiments examine the quality–latency trade-off of verification. Together, the results explain why improving marginal predictions and correcting the sampling procedure address distinct obstacles to reliable parallel generation.

1 INTRODUCTION

Autoregressive (AR) models are a dominant paradigm for discrete sequence generation, powering advances in natural language and biological sequence design (Achiam et al., 2023; Touvron et al., 2023; Madani et al., 2023). By decomposing a joint distribution into a chain of conditionals, they enable tractable training and high-fidelity generation. Yet standard decoding proceeds token by token, making sequential computation an inference bottleneck. Speculative decoding accelerates this process by proposing several tokens and verifying them against a reference model (Leviathan et al., 2023; Chen et al., 2023).

Masked diffusion models (MDMs) offer a different form of flexibility: they can reveal positions in arbitrary orders and predict several missing tokens in one model evaluation (Sahoo et al., 2024; Nie et al., 2025). Beyond throughput, any-order conditioning is attractive for globally structured generation. In code, infilling, and schema-constrained outputs, useful context need not lie to the left of the next token (Gong et al., 2025; 2024; Xiong et al., 2025). The same dependencies that make this flexibility useful, however, can make independent parallel decisions unreliable.

We distinguish two sources of this difficulty. *Order sensitivity* arises when model approximation errors accumulate differently along different generation paths. *Parallelization bias* arises when a sampler replaces a joint block conditional by independent token predictions: individually likely choices can form a jointly implausible sequence, even if every marginal is exact. The distinction matters because improving the predictor alone does not remove dependence discarded by the sampler. Figure 1 illustrates the two invariance baselines and their failures.

Confidence- and entropy-based schedules are widely used to decide which tokens to reveal (Chang et al., 2022; Peng et al., 2025; Ben-Hamu et al., 2025). They often choose both the order and the number of tokens committed together, making the two effects difficult to disentangle. This motivates

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.

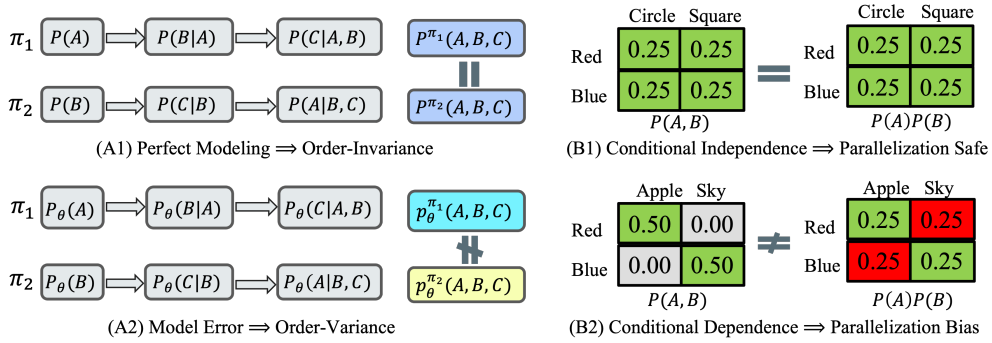

Figure 1: Two invariance baselines and their possible failures. Left: exact sequential conditionals give the same joint law under different reveal orders; imperfect conditional models can produce different laws. Right: independent marginals reproduce an independent joint, but can put mass on forbidden combinations when the target variables are dependent. Model imperfection permits order sensitivity; it does not require it.

our central questions: *why can order matter under an imperfect model, and what does it take to correct the sampling error introduced by parallel factorization?*

Contributions. Our analysis follows these questions. First, a rollout-weighted KL decomposition explains how local model errors and the contexts created by earlier decisions jointly determine order effects. Second, we characterize parallelization bias through forward KL, reverse KL, and incoherence, and relate exact whole-block rejection cost to conditional total correlation. Third, we study prefix-consistent verification as a practical relaxation: a Block-HMM isolates sampling error, while LLaDA and WeDLM experiments connect ordering and verification to task quality and measured latency. The contribution is this diagnostic synthesis and its implications for masked decoding, building on classical information-theoretic tools.

2 When Do Order and Parallelization Matter?

This section fixes notation and separates two idealized baselines from their possible failures (Figure 1). We consider a discrete sequence $X_{1:T} \in \mathcal{V}^T$ with target law $p = p_{\text{data}}$. Our focus is the distribution induced by decoding: the same trained model can produce different output laws under different generation procedures. In an MDM, *generation order* refers to the order in which positions are revealed and committed to the conditioning context, not necessarily their left-to-right order in the sequence. A sequential reveal order is a permutation π of the positions. A parallel schedule instead specifies an ordered partition $\mathcal{B} = (B_1, \ldots, B_M)$; blocks may be arbitrary subsets, not just contiguous spans. PCPD later uses a within-round verification order, which is distinct from the choice of blocks.

Perfect modeling: order invariance. Suppose the true one-token conditionals are available and generation is strictly sequential. For any fixed reveal order, the chain rule gives

$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_{\pi(t)} \mid x_{\pi(<t)}), \tag{1}$$

where $x_{\pi(<t)}$ is the context revealed before step t. Sampling each token from its conditional given this context therefore recovers the same joint law, whether positions are revealed left to right, right to left, or in another order. Left-to-right autoregressive sampling is one special case. This invariance concerns conditional sampling, not greedy argmax decoding.

Model error: order sensitivity. In practice, the decoder uses learned conditionals p_θ rather than the true conditionals of p. These predictions need not be compatible with a single joint law. Moreover, each prediction is evaluated on the context created by earlier model samples, not necessarily a context drawn from the data distribution. Changing the order changes these contexts and can therefore change the accumulated error. Thus order sensitivity is a property of the model-induced rollout, not of token positions alone; Section 3 makes this dependence explicit.

Conditional independence: exact parallelization. Let $P_m = \bigcup_{\ell<m} B_\ell$ be the positions revealed before a block. Exact joint block sampling remains valid by the chain rule: $p(x) = \prod_m p(x_{B_m} \mid x_{P_m})$. The usual parallel proposal instead samples the block's positions independently. To isolate this choice from model error, use true marginals:

$$p_{\prod}(x_{B_m} \mid x_{P_m}) = \prod_{i \in B_m} p(x_i \mid x_{P_m}), \qquad p_{\mathcal{B}}(x) = \prod_{m=1}^{M} p_{\prod}(x_{B_m} \mid x_{P_m}). \tag{2}$$

This proposal equals the true block conditional when the positions in B_m are conditionally independent given X_{P_m}. If this holds for every block, parallel commitment changes the computation but not the generated law: $p_{\mathcal{B}} = p$.

Conditional dependence: factorization bias. Structured sequences contain dependencies such as agreement, matching delimiters, and logical constraints. Two tokens can each be plausible given the current context yet become incompatible when combined. At a fixed context, the proposal in Equation (2) preserves every one-token marginal but discards these intra-block relationships. Consequently, even an oracle marginal predictor can induce sampling bias. The obstacle is not parallel computation itself, but replacing a dependent joint conditional with a product of marginals. Section 4 isolates this effect; learned parallel decoders combine it with model error, and changing the order or block size can affect both.

3 Order Sensitivity Under Model Error

For a fixed reveal order π, write $h = x_{\pi(<t)}$ for the revealed context. Replacing the exact conditionals in Equation (1) with learned predictions gives $p_\theta^\pi(x) = \prod_t p_\theta(x_{\pi(t)} \mid x_{\pi(<t)})$ as the law induced by that order.

Lemma 1 (Rollout-weighted KL chain rule)**.** *Define $\epsilon_j(h) = \mathrm{KL}(p_\theta(X_j \mid h) \| p(X_j \mid h))$. Then, for a fixed order and well-defined conditional kernels,*

$$\mathcal{J}(\pi) := \mathrm{KL}(p_\theta^\pi \| p) = \sum_{t=1}^{T} \mathbb{E}_{h \sim p_\theta^\pi(x_{\pi(<t)})}[\epsilon_{\pi(t)}(h)]. \tag{3}$$

Although the algebraic chain rule holds for every order, the expectation in Equation (3) is taken under that order's model-induced rollout. An early mistake therefore changes not only one token but also the contexts on which later conditional errors are evaluated. Adding and subtracting the data-prefix expectation separates local mismatch from this rollout-induced covariate shift.

Amplification of early errors. Under the additional assumption $0 \le \epsilon_j(h) \le B$, a TV bound and recursive prefix-distribution updates give

$$\mathcal{J}(\pi) \le \sum_{t=1}^{T} a_{\pi,t} + B \sum_{s=1}^{T-1} (T-s)\delta_{\pi,s}, \tag{4}$$

where $a_{\pi,t} = \mathbb{E}_{h\sim p}[\epsilon_{\pi(t)}(h)]$ and $\delta_{\pi,s} = \mathbb{E}_{h\sim p}[\mathrm{TV}(p_\theta(X_{\pi(s)} \mid h), p(X_{\pi(s)} \mid h))]$. The prefix expectation in each term uses the corresponding revealed positions. The proof is in Appendix A.

The coefficient $T - s$ captures a temporal asymmetry: a mismatch at step s can influence all later contexts. This gives a reason to favor reliable decisions early, while also accounting for how they change future predictions. Because changing order changes the local terms themselves, the bound is not an optimality result for a greedy ranking.

Connection to Easy-First decoding. The true conditional error is unavailable at inference. Easy-First uses conditional uncertainty as an accessible proxy and selects

$$\pi(t) = \arg\min_{j \notin \pi(<t)} H\big(p_\theta(X_j \mid x_{\pi(<t)})\big)\,. \tag{5}$$

Confidence or entropy can thus guide the choice of which context to establish first. The adaptive-policy extension is given in Appendix J.6. Teacher–student experiments support an entropy–error association, but not a calibrated error estimate (Appendix A.1). Low entropy can also reflect confident error, so the analysis does not assert universal Easy-First optimality or monotone gains with increasing model error.

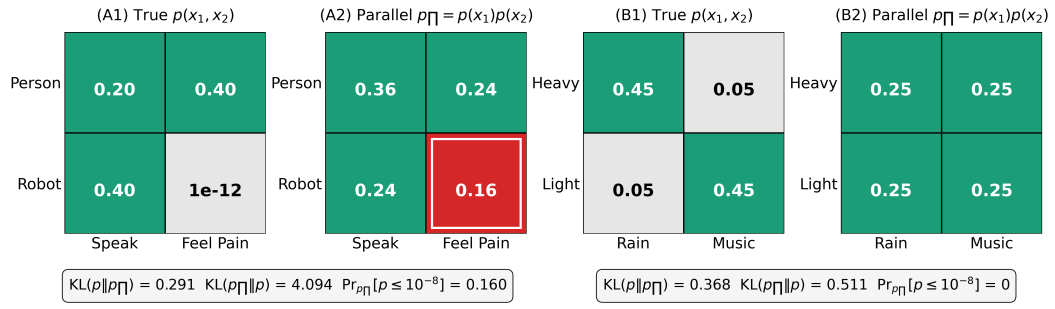

Figure 2: Complementary diagnostics for two illustrative joint laws. In A, factorization places about 0.16 mass on an entry with target probability approximately 10^{-12}. In B, dependence is lost without entering the displayed low-probability event. A has lower forward KL but much higher reverse KL and threshold-event probability than B. Values are in nats; displayed probabilities are rounded. The category labels are illustrative, not empirical semantic probabilities.

4 Sampling Error Induced by Parallel Generation

We now study the oracle factorized decoder $p_{\mathcal{B}}$ from Equation (2), with a fixed ordered partition $\mathcal{B}$. Using exact marginal conditionals removes model error, allowing us to ask how discarded intra-block dependence changes the generated joint law. Finite reverse-KL statements require support compatibility; target-zero prefixes and continued decoding are treated in Appendix B.

4.1 Three metrics for sampling error

These metrics answer different questions about the same decoder.

Forward KL: information loss. $\mathrm{KL}(p\|p_{\mathcal{B}})$ averages under the target distribution. It measures the dependence lost when a joint conditional is replaced by independent marginals.

Reverse KL: sampling risk. $\mathrm{KL}(p_{\mathcal{B}}\|p)$ averages under the decoder. It is sensitive to proposals that put mass on configurations with extremely small target probability, and becomes infinite under support mismatch.

Incoherence: event-level failure. For a specified failure set A, $p_{\mathcal{B}}(A)$ measures how often decoding fails. In the controlled study we use a target-likelihood threshold; on structured tasks, schema or execution failures are observable alternatives. Low sequence probability alone is not a semantic error.

These distinctions become explicit through the blockwise decompositions. Define forward and reverse conditional total correlation:

$$\overrightarrow{\mathcal{TC}}_m(h) = \mathrm{KL}\left(p(X_{B_m} \mid h) \,\middle\|\, \prod_{i \in B_m} p(X_i \mid h) \right), \tag{6}$$

$$\overleftarrow{\mathcal{TC}}_m(h) = \mathrm{KL}\left(\prod_{i \in B_m} p(X_i \mid h) \,\middle\|\, p(X_{B_m} \mid h) \right). \tag{7}$$

The chain rule gives

$$\mathrm{KL}(p\|p_{\mathcal{B}}) = \sum_m \mathbb{E}_{h\sim p}[\overrightarrow{\mathcal{TC}}_m(h)], \tag{8}$$

$$\mathrm{KL}(p_{\mathcal{B}}\|p) = \sum_m \mathbb{E}_{h\sim p_{\mathcal{B}}}[\overleftarrow{\mathcal{TC}}_m(h)]. \tag{9}$$

The directions differ in both the local divergence and the contexts used to evaluate it. In Figure 2, example A has less forward distortion than B but substantially greater reverse KL and failure probability. Thus forward KL alone need not rank decoders according to generated-sample risk. This does not make it uninformative: vanishing forward KL also controls event probabilities. Exact KL evaluation requires a known target law; the relationships and scope are detailed in Appendix J.

Combining model and dependence error. For a learned factorized block kernel $q_{\theta,m} = \prod_{i \in B_m} p_\theta(X_i \mid h)$, the forward divergence at a fixed context satisfies

$$\mathrm{KL}(p(X_{B_m} \mid h) \| q_{\theta,m}) = \overrightarrow{\mathcal{TC}}_m(h) + \sum_{i \in B_m} \mathrm{KL}(p(X_i \mid h) \| p_\theta(X_i \mid h)). \tag{10}$$

This identity (Appendix J.5) exposes the distinction directly: improving the learned marginals reduces the model-error terms, but the dependence term remains even when every marginal is exact. Changing the sampling rule is then needed to recover the target joint law. This also explains why ordering and parallelization interact: the chosen context and block determine both terms. A history-dependent block policy admits the corresponding path decomposition (Appendix J.7). The additive separation is forward-KL specific; it is not obtained by simply reversing both divergences.

4.2 Verification and the cost of exact correction

The preceding analysis turns a diagnostic question into an algorithmic one: how can a factorized proposal recover the dependencies it discards? In the Block-HMM's hard-parity limit, exact marginals still permit inconsistent blocks. Correct sampling therefore needs to account for joint dependence when forming, correcting, or accepting proposals. Verification against the known joint is one direct route. For a fixed independent proposal, rejection sampling makes the cost of this route explicit.

Proposition 1 (Whole-block rejection cost)**.** *Fix a context h and target block law $p(\cdot \mid h)$. Consider i.i.d. proposals from a fixed $q(\cdot \mid h)$ and a fixed acceptance function $a(x) \in [0,1]$ whose accepted law equals p. If $p \ll q$, then*

$$\mathbb{E}[N \mid h] \geq \sup_{x : p(x|h) > 0} \frac{p(x \mid h)}{q(x \mid h)} \geq \exp(\mathrm{KL}(p(\cdot \mid h) \| q(\cdot \mid h))) \,. \tag{11}$$

For $q = p_\prod$, the last term is $\exp(\overrightarrow{\mathcal{TC}}_m(h))$.

With acceptance rate Z, exactness requires $qa/Z = p$, so $Z \leq 1/\sup(p/q)$ and $\mathbb{E}[N] = 1/Z$ (Appendix B.2). The bound connects discarded dependence to the cost of accepting unchanged parallel proposals: accurate marginals do not by themselves make this correction inexpensive. It is specific to fixed whole-block rejection, not a lower bound on every exact sampler or on GPU latency. In particular, high reverse KL and expensive rejection are distinct phenomena (Appendix J).

From an oracle joint to a learned MDM. The controlled HMM supplies the joint probabilities needed for exact checks. A learned MDM normally supplies token marginals at a masked context. Checking whether a draft remains valid after earlier tokens are revealed requires additional conditional predictions. Unlike a causal AR evaluation, one bidirectional masked evaluation does not automatically provide every hypothetical-prefix conditional. This creates both an information gap for exact joint correction and a computational cost for practical verification. Reference-preserving diffusion speculation is possible (Gao et al., 2025; Wu & Zhang, 2025; Agrawal et al., 2025), but preserving a model decoder is a different guarantee from recovering the unknown data law.

Remasking as mitigation. Remasking revisits some decisions after observing others and can reduce inconsistency. Its effect depends on the update kernel: residual invalid mass gives infinite reverse KL under hard constraints, whereas suitable conditional kernels can preserve the target (Appendix F). The HMM experiment measures how much the specified factorized updates help, rather than treating all remasking procedures as equivalent.

4.3 Prefix-Consistent Parallel Decoding as a Relaxation

PCPD checks whether greedy parallel decisions survive conditioning on earlier proposed tokens (Figure 3). The same model supplies draft and verifier predictions. A round drafts the masked window, fixes an order, and verifies up to κ positions. It accepts the agreeing prefix, corrects the first mismatch, and recomputes the draft and order for the remaining positions. Independent hypothetical prefix contexts execute in one verifier batch.

PCPD makes one consequence of discarded dependence operational: a token that is confident under a masked context may change once correlated earlier decisions are fixed. The verifier tests this realized

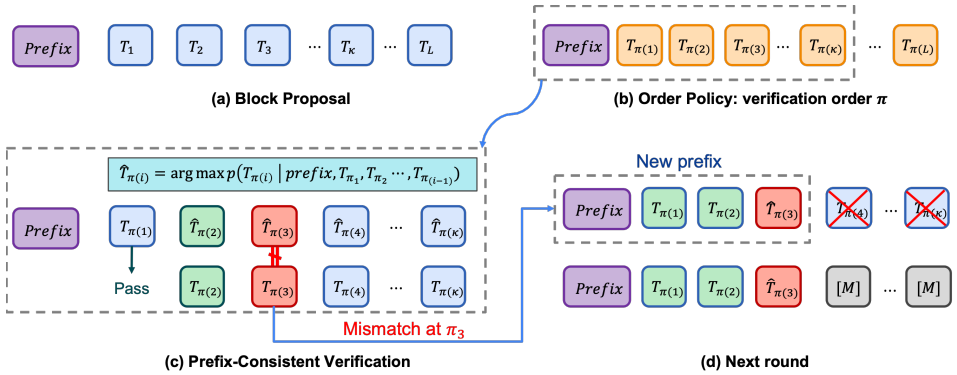

Figure 3: Prefix-consistent parallel decoding. The model drafts masked positions, chooses a within-round order, and batches speculative prefix contexts. At the first mismatch, it commits the verifier's token and discards the later speculative suffix. Those discarded contexts still contribute to actual computation.

prefix instability instead of simply applying another confidence threshold. With identical numerical evaluations and tie-breaking, batched checks reproduce their serial prefix verifier with the same order and round boundaries (Lemma 2).

This is a relaxation, not exact distributional correction. It need not reproduce a decoder that reranks after every token, and local agreement does not guarantee mode coverage or target-law recovery. The two-token counterexample, full algorithm, and compute accounting are given in Appendix C. The empirical question is whether these conditional checks improve the quality–latency trade-off enough to justify their additional work.

5 Experiments

We examine distributional error in a controlled Block-HMM, order and parallelization effects in language-model decisions, and the realized cost of prefix verification. Exact KL diagnostics use an accessible target law; the greedy language-model evaluations measure task quality instead.

5.1 Sampling error under parallel decoding in a Block-HMM

Why a Block-HMM? Structured outputs contain jointly constrained choices: a valid field value or code token in isolation need not remain valid with other choices. The Block-HMM gives a minimal, analytically accessible version of this issue. Because its joint and marginals are known, failures can be attributed to the sampling procedure without invoking a weak learned predictor.

Parity emissions and decoding. A two-state hidden Markov chain controls the bias of seven Bernoulli content bits in each block. An eighth bit is their XOR parity, flipped with probability η. As η decreases, parity-inconsistent configurations become increasingly unlikely while individual bits remain plausible. We compare sampling from exact marginals, factorized remasking, and exact joint block sampling. The last is an oracle reference implemented by enumeration, not a measured rejection algorithm. Appendix H specifies the update kernel and Monte Carlo estimators.

Results and interpretation. Figure 4 isolates the failure predicted by the analysis. At low parity noise, about half of independently generated blocks enter the specified low-probability region, despite exact marginal predictions. Reverse KL rises sharply as the constraint becomes sharper, while forward KL changes comparatively little. Remasking reduces these failures but leaves a gap for the tested update kernel. The experiment therefore separates three issues: marginal accuracy, joint consistency, and the effectiveness of a particular correction rule. It also shows why judging the proposal by forward KL alone can obscure its sensitivity to near-support violations.

Order diagnostics. The arithmetic study (Appendix G) retains the original comparison of L2R/R2L representations and masked reveal schedules. Carry propagation supplies a clear ordering structure,

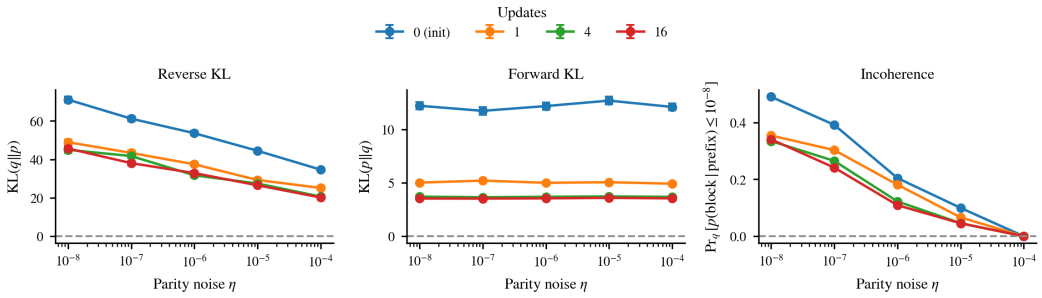

Figure 4: Controlled Block-HMM diagnostic ($B = 8$). Zero updates denotes the factorized proposal; other curves apply factorized remasking. KLs concern length-64 sequences; incoherence concerns individual blocks. Exact enumerated block sampling has zero KL, while low-probability events must still be interpreted relative to their target probability. The existing curves are Monte Carlo point estimates, not repeated-seed confidence bands.

Table 1: LLaDA greedy results. Differences are percentage points, not relative changes. GSM8K and HumanEval+ use LLaDA-8B-Base; JSON uses LLaDA-8B-Instruct. JSON here is a 200-schema subset, distinct from the WeDLM population. These runs have no repeated-run confidence intervals.

Task / metric	LC-256	LC-128	Difference
GSM8K accuracy ($n = 1319$)	71.72	66.57	-5.15
HumanEval+ pass@1 ($n = 164$)	28.66	18.90	-9.76
JSON schema validity ($n = 200$)	85.50	77.00	-8.50

and confidence-based LLaDA decoding performs well on this controlled task. The additional GSM8K ordering study extends the comparison to reasoning without an obvious fixed reveal order (Appendix D). Together with the entropy–error regressions, these experiments motivate uncertainty-based scheduling; they do not identify a universally optimal order or validate a universal error-amplification law.

5.2 LLaDA: quality under parallel commitment

We compare confidence-based LLaDA decoding at 256 and 128 steps for 256 generated tokens: one versus two commitments per step. Table 1 reports task quality; the step counts do not measure wall-clock speedup. Models, prompts, and block lengths differ across tasks as specified in Appendix D; each comparison uses its task's fixed protocol.

Committing two tokens together reduces quality across all three tasks, showing that fewer decoding steps can come at a substantial fidelity cost. Unlike the oracle HMM, these learned-model comparisons combine discarded dependence with model error and changing confidence trajectories. They demonstrate the practical trade-off, but do not establish that task difficulty measures conditional total correlation or predicts acceleration at matched quality.

5.3 Verification case study on WeDLM

We evaluate WeDLM-7B-Instruct (Liu et al., 2025a) on GSM8K, HumanEval+, and JSON-SchemaBench (Cobbe et al., 2021; Chen et al., 2021; Liu et al., 2023; Geng et al., 2025). PCPD uses the same model for proposal and verification, greedy decoding, and chunk size $\kappa = 9$. We compare with same-order one-token decoding, native WeDLM, and the same-model EB selection rule (Ben-Hamu et al., 2025). The distance penalty is selected on GSM8K training examples and fixed across tasks; the native and EB baselines use different selection policies. Measurements on RTX 6000 Ada include all verification work. The populations, concise settings, and calibration limitations are given in Appendix E.

Table 2: Measured total latency and primary quality (%). PCPD's penalty .5 is selected on GSM8K training data and fixed across tasks. The one-token reference has the same penalty. Native and EB rows are accelerated baselines with different selection policies.

Task	Decoder	Time (s)	Quality
GSM8K	One-token, penalty .5	4721.72	78.39
	WeDLM default	2247.88	76.57
	EB rule $\gamma = .2$	1902.70	77.56
	PCPD, penalty .5	1833.39	78.17
	EB rule $\gamma = .5$	1579.91	74.60
HumanEval+	One-token, penalty .5	458.37	70.73
	WeDLM default	115.04	65.85
	EB rule $\gamma = .2$	107.18	62.20
	PCPD, penalty .5	141.45	71.34
JSON schemas	One-token, penalty .5	1898.11	90.74
	WeDLM default	1030.71	85.47
	EB rule $\gamma = .2$	864.67	81.86
	PCPD, penalty .5	848.25	90.13

Quality and order must be compared together. Table 2 shows roughly 2–3$\times$ acceleration over same-order one-token decoding, with similar observed task quality. Against native WeDLM, the trade-off is task-dependent: PCPD is faster on GSM8K and JSON, while its higher HumanEval+ quality comes with longer latency. These comparisons also change the ordering policy, whose influence is visible in the penalty sensitivity (Appendix E). Thus not every gain should be attributed to verification, and similar aggregate scores do not establish output or distributional equivalence.

Verification under a fixed proposal policy. An additional ablation holds the aggressive EB proposal policy fixed and disables or enables PCPD (Appendix E). Verification recovers much of the quality lost under aggressive parallel commitment, but adds latency and computation relative to the unverified proposal. Compared with conservative EB, its benefit remains task-dependent. This supports conditional re-evaluation as a useful correction mechanism, not a cost-free or distributionally exact one.

Computational cost and transfer. PCPD reduces sequential calls while evaluating more conditional contexts and using additional memory. Its latency benefit therefore depends on efficient batching and prefix reuse, not simply on the number of rounds. Consistent with this distinction, the dense LLaDA diagnostic is slower than one-token decoding despite fewer sequential calls (Appendix D). The WeDLM results thus demonstrate an implementation-dependent trade-off, not a model-independent speedup.

6 Related Work

Dependence and schedules. Discrete Copula Diffusion addresses dependencies omitted by factorized outputs (Liu et al., 2025b). EB-Sampler decomposes model and joint dependence error and motivates adaptive entropy budgets (Ben-Hamu et al., 2025). ParallelBench supplies theory and targeted dependency-sensitive evaluations (Kang et al., 2025). Li and Cai analyze provable acceleration, and Chen et al. characterize schedule error through information curves and prove upper and lower bounds (Li & Cai, 2026; Chen et al., 2025). Our focus is the complementary diagnostic role of KL direction and incoherence, and its connection to the verification of realized proposals.

Dependency-aware acceleration. Fast-dLLM combines caching and confidence-based commitment (Wu et al., 2026); PUNT tests contextual independence (Azangulov et al., 2026); KLASS uses distributional changes (Kim et al., 2026). DeMask learns dependency scores, with its theoretical pairwise-to-joint control requiring a subadditivity assumption (Ringel et al., 2026). ADJUST and adaptive parallel decoding introduce richer joint proposals (Bansal & Sanghavi, 2025; Israel et al., 2026). Richer proposals, auxiliary computation, revisions, and altered masking schemes (Jiang et al., 2026; Yoo et al., 2026) change the sampler class; our rejection analysis concerns fixed factorized proposals.

Speculation and verification. Blockwise and autoregressive speculative decoding establish the draft–verify approach (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). SSD, FreeDave, and Spiffy apply related ideas to diffusion (Gao et al., 2025; Wu & Zhang, 2025; Agrawal et al., 2025). The distinction relevant here is between preserving a specified transition law, enforcing local greedy consistency, and sampling from the data law. These are different guarantees, not competing descriptions of one guarantee.

7 Conclusion

Generation order and parallel factorization expose different obstacles to reliable masked decoding. Model errors are evaluated on the contexts created by the model itself, whereas factorized sampling can discard joint dependence even when every marginal prediction is correct. The three diagnostics and the controlled HMM make this distinction concrete. Verification offers a way to revisit inconsistent parallel decisions, but its guarantee and cost depend on what is checked and how the required conditionals are obtained. PCPD illustrates this practical trade-off. The central lesson is to improve the predictor and the sampler together: confidence in individual tokens is not a substitute for joint consistency.

Limitations. The order bound assumes bounded conditional error and does not prove Easy-First optimality. Exact KL diagnostics require an oracle target, and the rejection bound covers only fixed whole-block proposals. PCPD has no stochastic-fidelity or mode-coverage guarantee. The real-model evidence concerns greedy decoding, filtered JSON populations, and specific implementations, without repeated hardware timing trials. Establishing a causal relation between dependence strength and useful acceleration on natural tasks requires further controlled study.

Ethics Statement

This work studies decoding reliability and efficiency using synthetic tasks, existing models, and public benchmarks. It does not involve human-subject research or collection of new personal data. More efficient generation can also amplify harmful or unreliable outputs; prefix consistency and task-level accuracy should not be interpreted as safety or factuality guarantees. External assets remain subject to their original licenses and terms of use.

Reproducibility Statement

The appendices specify assumptions and proofs, decoding and scoring protocols, model settings, and runtime accounting. An anonymous implementation is provided in Appendix I. The experimental sections distinguish distributional diagnostics from task-level quality and measured decoding latency.

AI Use Statement

AI assistance was used for manuscript editing, literature discovery, code inspection, figure scripting, and analysis of stored experimental outcomes. This assistance does not replace author verification of the methods, references, or reported results.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Christopher Lott, Fatih Porikli, and Mingu Lee. Structuring the future: Diffusion llm speculative decoding via calibrated draft graphs. *arXiv preprint arXiv:2509.18085*, 2025.

Iskander Azangulov, Teodora Pandeva, Niranjani Prasad, Javier Zazo, and Sushrut Karmalkar. Parallel sampling from masked diffusion models via conditional independence testing. In *International Conference on Learning Representations*, volume 2026, pp. 140485–140521, 2026.

Parikshit Bansal and Sujay Sanghavi. Enabling approximate joint sampling in diffusion lms. *arXiv preprint arXiv:2509.22738*, 2025.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

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

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. *arXiv preprint arXiv:2511.04647*, 2025.

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

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. *arXiv preprint arXiv:2510.04147*, 2025.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. Generating structured outputs from language models: Benchmark and studies. *arXiv e-prints*, pp. arXiv–2501, 2025.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*, 2024.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888, 2026.

Haozhe Jiang, Nika Haghtalab, and Lijie Chen. Diffusion language models are provably optimal parallel samplers. In *International Conference on Learning Representations*, volume 2026, pp. 28189–28203, 2026.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, et al. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. *arXiv preprint arXiv:2510.04767*, 2025.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models. *Advances in Neural Information Processing Systems*, 38:92267–92301, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Gen Li and Changxiao Cai. Breaking ar's sampling bottleneck: Provable acceleration via diffusion language models. *Advances in Neural Information Processing Systems*, 38:11700–11725, 2026.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025a.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *International Conference on Learning Representations*, volume 2025, pp. 88953–88979, 2025b.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8): 1099–1106, 2023.

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. *arXiv preprint arXiv:2502.09992*, 2025.

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

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

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

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Shutong Wu and Jiawei Zhang. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. *arXiv preprint arXiv:2510.00294*, 2025.

Zhen Xiong, Yujun Cai, Zhecheng Li, and Yiwei Wang. Unveiling the potential of diffusion large language model in controllable generation. *arXiv preprint arXiv:2507.04504*, 2025.

Jaehoon Yoo, Wonjung Kim, Chanhyuk Lee, and Seunghoon Hong. Infinite mask diffusion for few-step distillation. *arXiv preprint arXiv:2605.10518*, 2026.

A Order Dependence and Error Amplification Under Model Error

This section gives a more explicit derivation of why generation order matters under imperfect conditional models, and why early conditional errors are more consequential under autoregressive rollout.

For a generation order π, define the model-induced and data-induced prefix distributions at step t as

$$q_{\pi,t}(h) := p_\theta^\pi\big(x_{\pi(<t)} = h\big)\,, \qquad r_{\pi,t}(h) := p_{\mathrm{data}}\big(x_{\pi(<t)} = h\big)\,. \tag{12}$$

For a candidate variable X_j and prefix h, define the local conditional model error

$$\epsilon_j(h) := \mathrm{KL}\big(p_\theta(X_j \mid h) \,\big\|\, p_{\mathrm{data}}(X_j \mid h)\big)\,. \tag{13}$$

Then Lemma 1 gives

$$\mathcal{J}(\pi) := \mathrm{KL}\big(p_\theta^\pi \,\big\|\, p_{\mathrm{data}}\big) = \sum_{t=1}^{T} \mathbb{E}_{h\sim q_{\pi,t}}\big[\epsilon_{\pi(t)}(h)\big]\,. \tag{14}$$

Rollout-induced covariate shift. The prefix distributions evolve recursively as

$$q_{\pi,t+1}(h,x) = q_{\pi,t}(h)\, p_\theta\big(x_{\pi(t)} = x \mid h\big)\,, \tag{15}$$

$$r_{\pi,t+1}(h,x) = r_{\pi,t}(h)\, p_{\mathrm{data}}\big(x_{\pi(t)} = x \mid h\big)\,. \tag{16}$$

Thus, whenever the conditional model differs from the data conditional at an early step, the model-induced prefix distribution $q_{\pi,t}$ deviates from the data prefix distribution $r_{\pi,t}$. This deviation affects all later terms in Equation (14).

To separate local conditional error from rollout-induced distribution shift, we add and subtract the data-prefix expectation:

$$\mathcal{J}(\pi) = \sum_{t=1}^{T} \mathbb{E}_{h\sim r_{\pi,t}}\big[\epsilon_{\pi(t)}(h)\big] + \sum_{t=1}^{T} \big(\mathbb{E}_{h\sim q_{\pi,t}} - \mathbb{E}_{h\sim r_{\pi,t}}\big)\big[\epsilon_{\pi(t)}(h)\big]\,. \tag{17}$$

The first term measures local conditional mismatch under data prefixes. The second term captures rollout-induced covariate shift. This second term is order-dependent because $q_{\pi,t}$ is generated by the sequence of previous model conditionals in the order π.

If the conditional model were perfect, then $p_\theta(X_{\pi(t)} \mid h) = p_{\mathrm{data}}(X_{\pi(t)} \mid h)$ for all t and h. In that case $q_{\pi,t} = r_{\pi,t}$ for all t, the covariate-shift term vanishes, and the induced sampled joint law matches the data distribution for every order. Imperfection can therefore permit order sensitivity, but does not imply it: conditionals of a fixed incorrect joint law remain order-invariant under sampling. Greedy decoding has a different guarantee and can be order-sensitive even with exact conditionals.

Early-error amplification. We next make the amplification effect explicit. Suppose the local conditional errors are bounded, $0 \le \epsilon_j(h) \le B$, for all j and h. Then

$$\mathcal{J}(\pi) \le \sum_{t=1}^{T} \mathbb{E}_{h\sim r_{\pi,t}}\big[\epsilon_{\pi(t)}(h)\big] + B \sum_{t=1}^{T} \mathrm{TV}(q_{\pi,t}, r_{\pi,t})\,, \tag{18}$$

where $\mathrm{TV}(\cdot,\cdot)$ denotes total variation distance. The factor is B, not $2B$, because for any $f \in [0,B]$, $|\mathbb{E}_q f - \mathbb{E}_r f| \le B\,\mathrm{TV}(q,r)$: subtracting a constant from f does not change the expectation difference, and choosing the midpoint of the range gives $\|f - c\|_\infty \le B/2$, hence $|\sum_h (q(h)-r(h))(f(h)-c)| \le (B/2)\|q-r\|_1 = B\,\mathrm{TV}(q,r)$.

Define the one-step conditional mismatch at step s as

$$\delta_{\pi,s} := \mathbb{E}_{h\sim r_{\pi,s}}\big[\mathrm{TV}\big(p_\theta(X_{\pi(s)} \mid h), p_{\mathrm{data}}(X_{\pi(s)} \mid h)\big)\big]\,. \tag{19}$$

Using the recursive rollout relations in Equations (15) and (16), a standard sequential coupling argument gives

$$\mathrm{TV}(q_{\pi,t}, r_{\pi,t}) \le \sum_{s<t} \delta_{\pi,s}. \tag{20}$$

Table 3: Direct entropy–error regression on LLaDA teacher/student masked decoding. $\beta(H_t)$ is the standardized entropy coefficient.

	temp = 0		temp = 1	
Order	R^2	$\beta(H_t)$	R^2	$\beta(H_t)$
Easy-First	0.108	0.330	**0.412**	**0.644**
Random	0.168	0.415	0.312	0.490
Left-to-Right	0.122	0.331	0.244	0.422

Substituting Equation (20) into Equation (18) yields

$$\mathcal{J}(\pi) \leq \sum_{t=1}^{T} a_{\pi,t} + B \sum_{s=1}^{T-1} (T-s)\, \delta_{\pi,s}, \tag{21}$$

where

$$a_{\pi,t} := \mathbb{E}_{h\sim r_{\pi,t}} \left[\epsilon_{\pi(t)}(h) \right]. \tag{22}$$

The coefficient $(T-s)$ in Equation (21) captures the temporal asymmetry of autoregressive rollout: a mismatch at step s can influence the prefix distributions of all later steps $t > s$. Therefore, earlier conditional errors receive a larger downstream amplification factor.

Moreover, by Pinsker's inequality,

$$\mathrm{TV}\big(p_\theta(X_{\pi(s)} \mid h), p_{\mathrm{data}}(X_{\pi(s)} \mid h)\big) \leq \sqrt{\frac{1}{2}\epsilon_{\pi(s)}(h)}. \tag{23}$$

Thus local conditional KL controls an upper bound on the one-step TV mismatch. Early mismatches have larger coefficients in this upper bound; this is not a monotonicity or optimal-order theorem because changing order also changes the conditional errors. A finite alphabet alone does not guarantee a finite uniform B when target probabilities can be zero.

A.1 Entropy as an Empirical Error Proxy

Connection to Easy-First. The bound in Equation (21) motivates investigating a low-error-first heuristic, but does not prove that greedy ordering minimizes the bound or actual error. The true local error is unavailable at test time, so the practical decoding rule requires a proxy.

We directly test this proxy in a teacher–student masked-decoding experiment. The teacher is FP16 LLaDA-8B-Instruct and the deliberately degraded student is its GPTQ-3bit quantization. We decode 50 WikiText-2 sequences of 128 tokens for 64 unmasking steps, filling the student's selected token at each step. For each selected position t, we record teacher entropy H_t, teacher–student error

$$e_t = \mathrm{KL}(p_{\mathrm{teacher}}(\cdot \mid x_{\mathrm{masked}}) \parallel p_{\mathrm{student}}(\cdot \mid x_{\mathrm{masked}})),$$

and the mask ratio. We fit $e_t = \alpha H_t + \lambda \bar{e}_{<t} + b + \eta_t$ separately for Easy-First, random, and left-to-right commitment orders at temperatures 0 and 1. Table 3 reports the fit and Figure 5 shows all temperature-1 token instances.

The positive standardized coefficients 0.330–0.644 show that entropy contains a consistent local-error signal, while $R^2 = 0.108$–0.412 also shows that it is only a partial proxy. This experiment uses $\mathrm{KL}(p_{\mathrm{teacher}} \| p_{\mathrm{student}})$, whereas the rollout decomposition defines the reverse local direction $\mathrm{KL}(p_\theta \| p_{\mathrm{data}})$; we use it to validate association with approximation difficulty, not to claim equality with the theoretical local-error term. At deployment the teacher entropy is also unavailable, and the decoder substitutes its own predictive entropy or confidence. These two gaps are why Easy-First is presented as an empirically supported heuristic.

As a direction-matched auxiliary test, we also use a Qwen2.5-7B teacher and a Qwen2.5-0.5B student on 100 GSM8K prompts. We follow each student's temperature-zero rollout for 200 tokens and regress $\mathrm{KL}(p_{\mathrm{student}} \| p_{\mathrm{teacher}})$ on student conditional entropy and mean past error. For prefixes of length 20, 100, and 200, respectively, R^2 is $0.197, 0.145, 0.164$ and the standardized entropy

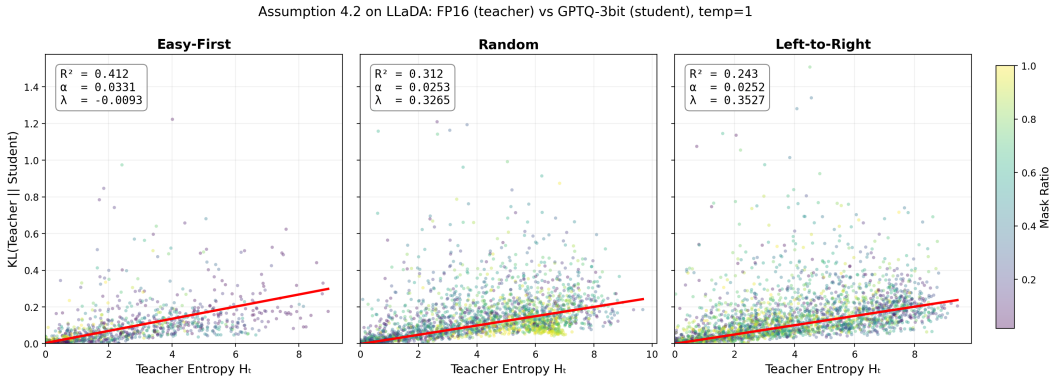

Figure 5: Teacher conditional entropy versus local teacher–student KL on LLaDA at temperature 1. Each point is a selected token and color denotes the mask ratio. Entropy is a meaningful but incomplete predictor; the relationship varies with the commitment order.

coefficient is $0.444, 0.362, 0.393$. All entropy coefficients remain significant with sequence-cluster-robust standard errors ($p < 2.1 \times 10^{-41}$). Unlike the MDM experiment above, this stress test matches the theoretical KL direction and uses entropy from the approximate model itself, but it is autoregressive and uses a larger model as a proxy for the unknown data conditional. We therefore regard the two tests as complementary evidence for correlation, not calibration.

Using conditional entropy as a ranking signal gives the Easy-First rule

$$\pi(t) = \operatorname*{argmin}_{j \notin \pi(<t)} H\left(X_j \mid x_{\pi(<t)}\right). \tag{24}$$

This does not claim that entropy fully determines local model error. Rather, entropy serves as a practical uncertainty signal for approximating the low-error-first ordering suggested by Equation (21). The bound identifies how early errors can influence more future contexts. It does not imply that an entropy-based policy's advantage increases monotonically with model error. The regression is not used as an assumption in either the exact KL identity or the bounded-error upper bound; in particular, no exponential weighting based on a fitted propagation coefficient is needed for those results.

B Proofs for Sampling Error and Verification Cost

B.1 Blockwise KL decompositions

Write $p_m = p(X_{B_m} \mid X_{P_m})$ and $q_m = p_{\prod}(X_{B_m} \mid X_{P_m})$. For $p = \prod_m p_m$ and $q = p_{\mathcal{B}} = \prod_m q_m$, expanding the log ratio and conditioning on each prefix gives

$$\mathrm{KL}(p\|q) = \mathbb{E}_p \sum_m \log(p_m/q_m) = \sum_m \mathbb{E}_{X_{P_m}\sim p}\mathrm{KL}(p_m\|q_m), \tag{25}$$

$$\mathrm{KL}(q\|p) = \mathbb{E}_q \sum_m \log(q_m/p_m) = \sum_m \mathbb{E}_{X_{P_m}\sim q}\mathrm{KL}(q_m\|p_m). \tag{26}$$

For products of exact marginals, the inner terms are respectively forward and reverse conditional total correlation. The prefix measures differ. Finite identities require the stated support conditions; if decoding reaches a target-zero prefix, specify subsequent kernels and use extended-real KL.

B.2 Whole-block rejection bound

Fix a context and suppress it in the notation. Let q be the fixed proposal and $a(x) \in [0,1]$ its acceptance function, with acceptance rate $Z > 0$. Exactness requires $q(x)a(x)/Z = p(x)$. Thus

$$Z \le \frac{1}{M}, \qquad M = \sup_{x:p(x)>0} \frac{p(x)}{q(x)}, \qquad \mathbb{E}[N] = Z^{-1} \ge M. \tag{27}$$

Here N is geometric because proposals and the acceptance rule are fixed. Since $\log M \geq \log(p(x)/q(x))$ on target support,

$$\log M \geq \mathbb{E}_p \log(p/q) = \mathrm{KL}(p\|q). \tag{28}$$

For $q = \prod_i p_i$, this yields $\mathbb{E}[N] \geq \exp(\overrightarrow{\mathcal{TC}})$, proving Proposition 1. If $p \not\ll q$, exact unchanged-proposal rejection is impossible. This argument does not cover partial acceptance, corrected proposals, or arbitrary exact samplers.

B.3 Support mismatch and observable events

If $q(x) > 0$ and $p(x) = 0$, the corresponding term in $\sum_x q(x)\log(q(x)/p(x))$ is infinite. A blockwise violation occurring at a positive-probability decoder prefix therefore makes global reverse KL infinite by Equation (26).

For $A_\tau = \{x : p(x) \leq \tau\}$, $0 < \tau < 1$, Markov's inequality applied to $-\log p(X)$ under q gives

$$q(A_\tau) \leq \frac{\mathbb{E}_q[-\log p(X)]}{\log(1/\tau)} = \frac{\mathrm{KL}(q\|p) + H(q)}{\log(1/\tau)}. \tag{29}$$

The entropy term cannot be discarded. Event-level comparisons using both KL directions and target base rates are given in Appendix J.

C Prefix-Consistent Parallel Decoding: Algorithm and Guarantees

C.1 Prefix-consistency rule

Let $C \subseteq B$ be the committed positions and $M = B \setminus C$. The shared deterministic greedy draft/verifier rule $\mathcal{D}_0$ uses fixed tie-breaking. The draft and within-round order are

$$\tilde{x}_i = \mathcal{D}_0(p_\theta(X_i \mid x_P, x_C, \mathsf{mask}_M)), \quad i \in M, \qquad \pi = \Omega(x_P, x_C, \tilde{x}_M). \tag{30}$$

For rank j, the hypothetical prefix inserts the earlier draft tokens:

$$V_j = \mathcal{D}_0\big(p_\theta(X_{\pi_j} \mid x_P, x_C, \tilde{x}_{\pi_{<j}}, \mathsf{mask}_{M\setminus\{\pi_1,\ldots,\pi_{j-1}\}})\big). \tag{31}$$

These contexts can be evaluated in a batch. A prefix of length r is consistent when $\tilde{x}_{\pi_j} = V_j$ for all $j \leq r$. PCPD commits matching tokens and the first corrected token, then discards later speculative contexts. Agreement makes the hypothetical prefix equal to the realized prefix only up to the first mismatch. In Algorithm 1, both decoding operators equal $\mathcal{D}_0$.

C.2 Batched versus serial prefix verification

Lemma 2 (Batched prefix equivalence)**.** *With the same deterministic model evaluations, tie-breaking, within-round order, chunk size, and round boundaries, batched PCPD and its serial prefix verifier commit identical tokens.*

Proof. The rank-one context equals the draft context, so its draft is already the verifier's token. Inductively, until the first mismatch every earlier draft token is accepted; each batched context therefore equals the serial context. At the mismatch both methods commit the same corrected token and discard subsequent branches. If no mismatch occurs, both commit the whole chunk. They consequently start the next round from the same state, completing the induction. □

The numerical-evaluation assumption matters: different GPU batch kernels may change near-tied argmax decisions. Our implementation includes parity checks and a numerical guard. This lemma does not establish equivalence to a native decoder that reranks after every committed token.

Algorithm 1 Prefix-Consistent Parallel Decoding (PCPD)

Require: Context x_P, masked block B, model p_θ, mask token mask, order policy Ω, chunk size κ
Require: Deterministic draft rule $\mathcal{D}^{\mathrm{draft}}$ and verifier rule $\mathcal{D}^{\mathrm{verify}}$
Ensure: Decoded block x_B

```
Initialize C ← ∅ and x_i ← mask for all i ∈ B
while C ≠ B do
    Let M ← B \ C
    Compute draft tokens (x̃_i)_{i∈M} using D^draft in Equation (30)
    Choose order π = (π_1, ..., π_{|M|}) using Equation (30)
    Let r ← min(κ, |π|)
    Commit x_{π_1} ← x̃_{π_1} and update C ← C ∪ {π_1}
    Batch-evaluate verifier contexts for ranks j = 2, ..., r using draft prefixes x̃_{π_<j}
    for j = 2, ..., r do
        Obtain V_j using D^verify in Equation (31)
        if V_j = x̃_{π_j} then
            Commit x_{π_j} ← x̃_{π_j} and update C ← C ∪ {π_j}
        else
            Commit x_{π_j} ← V_j and update C ← C ∪ {π_j}
            Ignore later verifier outputs, leave later ranks masked, and break
        end if
    end for
end while
return x_B
```

C.3 Verification targets and limitations

Exact whole-block rejection preserves $p(X_B \mid x_P)$. Reference-preserving speculation instead preserves a specified model transition law; PCPD checks a narrower greedy prefix condition. In autoregressive speculation, one causal evaluation provides the hypothetical-prefix predictions. A standard bidirectional masked evaluation provides marginals at one context, so distinct prefixes generally require additional computation, specialized attention, or caching. SSD, FreeDave, and Spiffy exploit such structure for specified reference decoders (Gao et al., 2025; Wu & Zhang, 2025; Agrawal et al., 2025). Spiffy's block-tree exactness is architecture-dependent; its fully bidirectional implementation is described as near-lossless. Neither model-reference preservation nor local prefix agreement certifies correctness under the unknown data law.

Two-mode counterexample. Let $p(00) = p(11) = 1/2$. The product of marginals is uniform on four assignments, with invalid mass $1/2$ and $\mathrm{KL}(p\|p_{\prod}) = \log 2$. Exact unchanged-proposal rejection needs two draws on average. Greedy PCPD with ties resolved to zero instead outputs 00 every time: it is prefix-consistent but misses the other mode. Its changed proposals and deterministic target are outside Proposition 1.

Chunk size. Larger κ can reduce sequential calls but compute more discarded branches. It also changes when proposals and order are refreshed, so final outputs may change. Speed and task quality must therefore both be measured.

C.4 PCPD Compute and Memory Accounting

We make the best- and worst-case costs explicit. Consider a block with m masked positions and $\kappa \geq 2$. A proposal round commits at most κ tokens; it commits at least two unless only one position remains, because the first draft token and either the matching or corrected second token are both committed. Consequently, the number of proposal rounds R satisfies

$$\left\lceil \frac{m}{\kappa} \right\rceil \leq R \leq \left\lceil \frac{m}{2} \right\rceil . \tag{32}$$

The lower bound is attained when every verified chunk is consistent; the upper bound is attained when every non-singleton round mismatches at rank two. For $\kappa = 1$, PCPD reduces to one-token commitment and $R = m$.

Each non-singleton round makes one proposal call and one batched verifier call; a final singleton round needs only the proposal. Thus the best-case number of sequential model-call rounds is

$$2\left\lceil \frac{m}{\kappa} \right\rceil - \mathbf{1}\{m \bmod \kappa = 1\}, \tag{33}$$

while the worst case is exactly m, the same sequential-round count as one-token decoding. These are latency-oriented counts, not FLOP counts.

For context-row and memory accounting, a verifier call for a chunk of length r materializes $r-1$ separate prefix contexts in its batch. Counting the proposal as one model-input row, a round therefore materializes r rows. The best case materializes exactly m rows over the block. In the rank-two-mismatch worst case it materializes

$$\sum_{q=0}^{\lceil m/2 \rceil - 1} \min\{\kappa, m-2q\} \;\le\; \kappa \left\lceil \frac{m}{2} \right\rceil \tag{34}$$

rows, including branches discarded after the first mismatch. Peak verifier batch size is $\min\{\kappa, m\}-1$. Fewer sequential calls can therefore coexist with more materialized rows and larger peak memory. Actual FLOPs depend on context lengths and cached computation; row counts are not FLOP measurements. Latency depends on prefix stability and hardware execution of the verifier batch. We report these quantities separately.

D LLaDA Experimental Details

Models and evaluation. We use LLaDA-8B-Base for GSM8K and HumanEval+, and LLaDA-8B-Instruct for JSONSchemaBench, with bfloat16 inference and no additional model training. Evaluation covers 1,319 GSM8K test questions, all 164 HumanEval+ tasks, and 200 JSON schemas sampled equally from the trivial and easy subsets. GSM8K uses four-shot reasoning prompts; code generation uses the benchmark prompts; JSON prompts request a schema-compliant object. We report numerical answer accuracy, HumanEval+ pass@1, and JSON schema validity, respectively. JSON evaluation is restricted to object schemas of at most 3,000 characters without composition or negation; validity of an extracted object is distinguished from strict output parsing.

Decoding. All comparisons use greedy decoding, 256 generated tokens without early stopping, and no classifier-free guidance. Block lengths are 32, 256, and 128 for GSM8K, HumanEval+, and JSON, respectively. The low-confidence baselines commit the most confident predictions: 256 steps commit one token per step and 128 steps commit two. PCPD uses the same confidence ranking without a distance penalty and the correction rule in Algorithm 1. The selected chunk sizes are $\kappa = 32, 24, 6$ for the three tasks. They are descriptive operating points chosen from the reported task-level ablations, not from an independent tuning set; Figure 6 retains the comparison curves.

Timing and additional ordering comparison. The scheduling study used H100 NVL GPUs with serial prefix verification. The separate three-prompt timing diagnostic uses LLaDA-8B-Instruct on RTX 6000 Ada, with block and chunk size 32. It recomputes full bidirectional contexts without prefix-cache reuse and counts discarded verification work. The estimated batch rounds in Figure 6 are not measured latencies.

An additional four-shot study on 100 GSM8K questions compares confidence and random orders across FP16/GPTQ-3bit precision, temperatures zero/one, and 32–256 decoding steps. Confidence ordering performs better in these settings, but its advantage does not grow monotonically with model error: at temperature zero and 256 steps, quantization reduces the gap from eight to three percentage points.

E WeDLM Fully Batched Experimental Details

Models and evaluation. We use WeDLM-7B-Instruct with bfloat16 inference, greedy decoding, a 256-token limit, window size 16, and one request at a time. Evaluation covers 1,319 GSM8K test questions, 164 HumanEval+ tasks, and 1,803 JSON schemas (221 trivial and 1,582 easy). GSM8K uses zero-shot reasoning prompts and numerical answer accuracy; HumanEval+ measures success

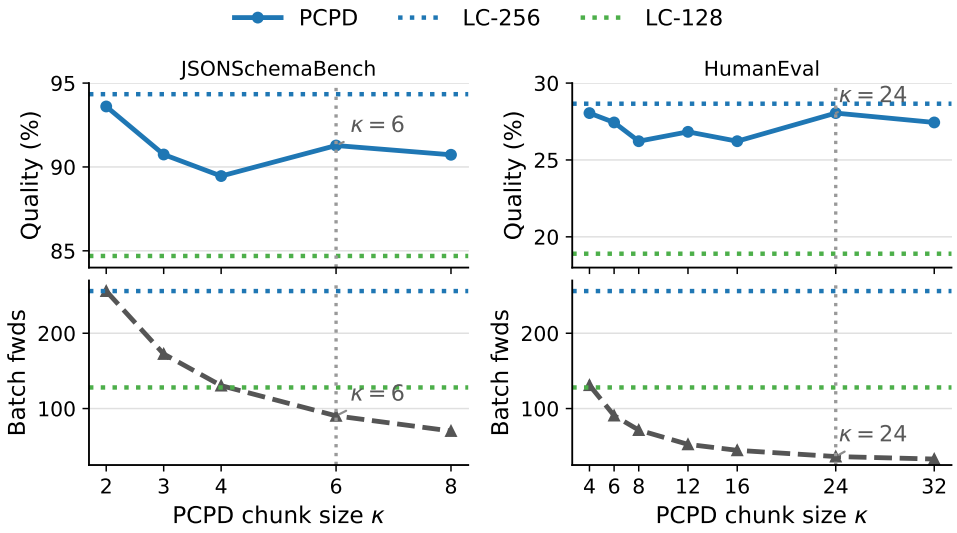

Figure 6: PCPD chunk-size ablation on LLaDA. Increasing κ verifies more ordered ranks per batched call, reducing batch-forward rounds, but can change the proposal/correction trajectory because the proposal and order are recomputed less often. The selected settings retain quality close to the 256-step baseline with fewer estimated batch rounds.

Table 4: PCPD chunk/window ablation on a fixed 128-example GSM8K subset.

Sweep	Setting	1-token s	PCPD s	Speedup	1-token EM	PCPD EM	McNemar p	Rows
κ	3	441.2	311.8	1.42×	80.47	81.25	1.000	1.05×
κ	5	441.6	210.2	2.10×	80.47	83.59	0.388	1.11×
κ	9	442.5	166.3	2.66×	80.47	81.25	1.000	1.37×
κ	13	440.7	203.1	2.17×	80.47	82.03	0.774	1.72×
window	8	415.8	155.5	2.67×	84.38	78.91	0.092	1.29×
window	16	442.5	166.3	2.66×	80.47	81.25	1.000	1.37×
window	32	477.6	251.5	1.90×	82.03	80.47	0.804	1.35×

on both base and additional tests. JSON evaluation uses the same schema restrictions as the LLaDA study and measures validity of extracted objects, not strict whole-output parsing.

Decoders and ablations. The one-token reference follows WeDLM's adjusted-entropy order but commits one position per evaluation. PCPD uses $\kappa = 9$ and distance penalty .5, selected on 128 GSM8K training examples and fixed across tasks. The native baseline uses threshold .4 and penalty .02. The same-model EB baseline selects the largest entropy-ranked prefix satisfying $\sum_{i=1}^{r} H_i - \max_{i \le r} H_i \le \gamma$. The matched EB+PCPD comparison fixes $\gamma = 5$, penalty .05, and maximum chunk 16, verifying every multi-token proposal. These settings were calibrated on the same training subset; the broader study also included exploratory test comparisons. The chunk/window ablation in Table 4 uses 128 test questions and penalty .02; it is diagnostic, not held-out tuning.

Order sensitivity. The main results depend on the ordering policy. For example, at penalty .02, PCPD obtains 64.63% HumanEval+ accuracy in 141.8 s, while native WeDLM at threshold .4 obtains 65.85% in 115.0 s. Thus verification does not uniformly dominate the native decoder.

Timing. Measurements use RTX 6000 Ada GPUs and CUDA-synchronized generation latency, excluding loading and warmup. Methods alternate execution order across examples. Reported totals sum per-example latencies, including discarded verification work; they are not elapsed times of parallel jobs. Batched verification is checked against its serial counterpart on 16 examples per task, with agreement on all 48 checks.

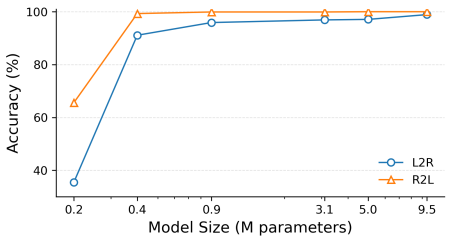

Figure 7: Comparison of L2R and R2L decoding strategies.

Table 5: **Unmasking Strategy Comparison in LLaDA (10-digit Addition).** Overall accuracy (%) with edge case accuracy in parentheses. Columns denote the number of denoising steps / total generation steps.

Strategy	16	32	64	128
Left-to-Right	76.0 (45.0)	69.0 (50.0)	79.0 (50.0)	84.0 (62.5)
Random	87.0 (80.0)	75.0 (77.5)	78.0 (75.0)	79.0 (80.0)
Right-to-Left	93.0 (82.5)	87.0 (85.0)	94.0 (85.0)	92.0 (80.0)
Low-Confidence	**97.0** (95.0)	**89.0** (80.0)	**99.0** (97.5)	**99.0** (97.5)

F Remasking and Residual Support Violations

Fix a context, with target support $\mathcal{S} = \{x : p(x) > 0\}$. Let $q^{(0)}$ be the initial block law and $K_{1:n}$ the composition of the first n remasking kernels. Then

$$q^{(n)}(v) = \sum_u q^{(0)}(u) K_{1:n}(v \mid u). \tag{35}$$

Proposition 2 (Residual invalid mass)**.** *On a finite state space,* $\mathrm{KL}(q^{(n)} \| p)$ *is finite if and only if* $K_{1:n}(\mathcal{S}^c \mid u) = 0$ *for every* $u \in \mathrm{supp}(q^{(0)})$.

Proof. All terms in the mixture are nonnegative. If a reachable u can transition to $v \notin \mathcal{S}$, then $q^{(n)}(v) \geq q^{(0)}(u) K_{1:n}(v \mid u) > 0$ and reverse KL is infinite. Otherwise $q^{(n)}$ is supported on $\mathcal{S}$; finiteness follows from the finite state space and $p(v) > 0$ there. □

Support containment is not sufficient for $q^{(n)} = p$. Conversely, exact conditional updates or suitable invariant kernels can preserve the target. This result is a conditional support criterion, not a general impossibility theorem for remasking. Under positive parity noise, the target has full support and KL need not be infinite; the residual errors of the particular factorized remasking kernel are measured in Figure 4. The kernel, random subset rule, and update counts are specified in Appendix H.

G Experimental Details for Arithmetic Tasks

This section provides detailed settings and results for the arithmetic reasoning experiments, with a focus on the 10-digit addition task. We describe the dataset construction, difficulty stratification, model configurations, training procedure, and evaluation protocol.

Carry propagation provides a computational reason to study right-to-left representations. However, conditional on both operands the correct sum is deterministic: true answer entropy is zero in either direction. The NanoAdder curves compare separately trained representations/models, not an intervention on reveal order in one fixed conditional model. They therefore do not isolate an order-by-model-error causal effect. In Table 5, confidence-based decoding reaches 99% accuracy, but accuracy is not monotone in the number of steps (e.g., 97% at 16 steps and 89% at 32). These are supplementary observations, not direct validation of a total-correlation mechanism.

Data and evaluation. The fixed synthetic suite contains 1,000 nonnegative addition problems: 10% with 1–3 digit operands, 20% with 4–6 digits, 30% with 7–9 digits, 20% with 10 digits, 10% edge cases (cascading carries, zeros, near-maximum values), and 10% mixed operand lengths. The same seeded examples are reused across methods. Accuracy is exact match of the complete integer result; R2L outputs are reversed before comparison. Edge-case accuracy is also reported in Table 5.

NanoAdder training. The separately trained autoregressive models use a character vocabulary (digits, +, =, padding), context length 48, learned token and positional embeddings, and causal Transformer encoder stacks at the scales in Figure 7. On-the-fly training samples operand lengths uniformly up to 10 digits. Next-token loss applies only after =. Training uses AdamW, cosine learning-rate decay, linear warmup, gradient clipping, 50,000 steps, and batch size 256. These representation comparisons are not fixed-model reveal-order interventions.

H Experimental Details for the Block-HMM Study

This appendix provides implementation-level details for the Block-HMM experiments in Section 5.1.

H.1 Model specification

We study sampling error under parallel decoding in a fully specified generative model, where the target distribution is known exactly and no model approximation error is present. The data distribution is defined by a Block-HMM with parity emissions.

Latent dynamics. Let $Z_n \in \{1, \dots, K\}$ denote the latent state associated with block n. The latent states follow a first-order Markov chain

$$Z_1 \sim \pi, \qquad Z_n \mid Z_{n-1} \sim A. \tag{36}$$

We use a symmetric transition matrix parameterized by a single self-transition probability a:

$$A_{zz} = a, \qquad A_{zz'} = \frac{1-a}{K-1} \quad (z' \neq z). \tag{37}$$

Unless otherwise stated, we use $K = 2$, $\pi = (0.5, 0.5)$, and $a = 0.9$, yielding persistent latent modes across blocks.

H.2 Block emission distribution

Each block $X^{(n)} \in \{0,1\}^B$ consists of one parity bit and $B-1$ content bits,

$$X^{(n)} = (P_n, Y_{n,1}, \dots, Y_{n,B-1}). \tag{38}$$

Conditioned on the latent state $Z_n = z$, the content bits are independent Bernoulli variables,

$$Y_{n,i} \mid Z_n = z \sim \mathrm{Bern}(\rho_z), \qquad i = 1, \dots, B-1, \tag{39}$$

where $\{\rho_z\}_{z=1}^K$ are fixed, state-specific parameters shared across all blocks. In our experiments, we use

$$\rho = (0.9, 0.1).$$

The parity bit enforces a global XOR constraint with noise,

$$E_n = \mathrm{Bern}(\eta), \qquad P_n = \bigoplus_{i=1}^{B-1} Y_{n,i} \oplus E_n. \tag{40}$$

As $\eta \to 0$, the model induces strong intra-block dependence and near-zero-support configurations.

H.3 Exact conditional distributions

Given a prefix $x^{<n}$ ending at a block boundary, the true conditional distribution of the next block is

$$p(x^{(n)} \mid x^{<n}) = \sum_z p(z_n = z \mid x^{<n})\, p(x^{(n)} \mid Z_n = z). \tag{41}$$

The posterior $p(z_n \mid x^{<n})$ is computed exactly using standard HMM forward filtering. For a fixed block size B, the conditional distribution $p(x^{(n)} \mid x^{<n})$ is obtained by explicit enumeration over all 2^B possible block configurations.

H.4 Decoding procedures

We compare three decoding strategies.

Mean-field parallel decoding. At each block boundary, we compute the exact marginal probabilities

$$p(x_{n,i} = 1 \mid x^{<n}), \quad i = 1, \dots, B,$$

and sample each bit independently according to these marginals. This preserves all marginal conditionals but discards intra-block dependencies.

Iterative remasking. Remasking initializes the block distribution from the mean-field proposal and then repeatedly selects a subset of positions to resample while keeping the remaining positions fixed. In each update, the resampled positions use exact conditional marginals under the true block conditional given the kept values, but they are still resampled independently. For $B = 8$, we remask 4 positions per update. The remasked subset is selected uniformly among all $\binom{8}{4}$ subsets of size 4 at every update. In the implementation, we compute the induced distribution by explicitly averaging over all such subsets, which is equivalent to the expectation of this uniformly random selection policy. We report update counts $\{0, 1, 4, 16\}$, where 0 denotes the initial mean-field proposal.

Exact block sampling (oracle baseline). As a control, we sample each block directly from the true conditional distribution $p(x^{(n)} \mid x^{<n})$. This is implemented by enumerating all 2^B block configurations and sampling according to their exact conditional probabilities. This procedure produces exact samples from the target distribution and serves solely as a zero–sampling-error reference. It is not an implemented rejection verifier or a latency comparison.

H.5 Evaluation metrics

We evaluate decoding quality using three complementary metrics.

Reverse KL divergence.

$$\mathrm{KL}(q\|p) = \mathbb{E}_{x\sim q}[\log q(x) - \log p(x)], \tag{42}$$

estimated via Monte Carlo sampling from the decoding distribution. For each parity-noise level and decoder setting, we use 300 independently generated length-64 sequences.

Forward KL divergence.

$$\mathrm{KL}(p\|q) = \mathbb{E}_{x\sim p}[\log p(x) - \log q(x)], \tag{43}$$

estimated via Monte Carlo sampling using the verified decoder. We again use 300 independently generated length-64 target samples for each setting.

Incoherence rate. We report the fraction of blocks whose true conditional probability satisfies

$$p(x^{(n)} \mid x^{<n}) \le \tau, \tag{44}$$

with threshold $\tau = 10^{-8}$.

H.6 Parameter settings

Unless otherwise stated, we use the following parameter configuration:

- Number of latent states: $K = 2$
- Block size: $B = 8$
- Sequence length: $T = 64$
- Initial distribution: $\pi = (0.5, 0.5)$
- Self-transition probability: $a = 0.9$
- Bernoulli parameters: $\rho = (0.9, 0.1)$
- Parity noise: $\eta \in \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$
- Incoherence threshold: $\tau = 10^{-8}$
- Remasking updates: $\{0, 1, 4, 16\}$ with 4 uniformly selected remasked positions per update
- Monte Carlo samples: 300 sequences for reverse KL/incoherence and 300 verified target sequences for forward KL, per parity-noise level and decoder setting

All KL values are estimated using Monte Carlo averages over independently generated sequences. Each length-64 sequence contains $64/8 = 8$ blocks, so the reported incoherence rate averages over 2400 block-level events per setting.

Scope of the structured-text analogy. The Block-HMM is a controlled diagnostic, not a realistic language model. Its known joint and exact marginals isolate factorization error, its parity noise tunes near-support risk, and its latent state adds cross-block context. Structured outputs also contain jointly constrained fields or tokens, but this analogy does not establish that their observed errors arise from the same mechanism. Task accuracy and schema validity do not estimate the unknown data-distribution KL.

I Code Availability and Resources

The core implementation and evaluation utilities are available at `https://anonymous.4open.science/r/pcpd_hmm-3331`. External models and datasets retain their original licenses and are not redistributed. The code distribution preserves upstream attribution and the complete WeDLM license terms.

The Block-HMM experiments and arithmetic data generation used CPUs. The LLaDA scheduling study used H100 NVL GPUs; WeDLM and the separate dense LLaDA timing diagnostic used RTX 6000 Ada GPUs.

J Scope and Diagnostic Counterexamples

All logarithms in this appendix are natural. The constructions below are elementary checks on the interpretation of the analysis, not claimed as new general sampling lower bounds.

J.1 Noisy parity: risk is not rejection cost

For $b \geq 3$ bits and $0 < \eta \leq 1/2$, define

$$p_\eta(x) = \begin{cases} (1-\eta)/2^{b-1}, & \bigoplus_i x_i = 0, \\ \eta/2^{b-1}, & \bigoplus_i x_i = 1. \end{cases} \tag{45}$$

Every proper-subset marginal is uniform, so the product of the univariate marginals is $q(x) = 2^{-b}$. Direct calculation gives

$$\mathrm{KL}(p_\eta \| q) = \log 2 - h(\eta), \tag{46}$$

$$\mathrm{KL}(q \| p_\eta) = -\tfrac{1}{2}\log(4\eta(1-\eta)), \tag{47}$$

$$q(\text{odd parity}) = \tfrac{1}{2}, \qquad p_\eta(\text{odd parity}) = \eta, \tag{48}$$

$$\sup_x p_\eta(x)/q(x) = 2(1-\eta), \tag{49}$$

where $h(\eta) = -\eta\log\eta - (1-\eta)\log(1-\eta)$. As $\eta \downarrow 0$, reverse KL diverges while forward KL tends to $\log 2$, and optimal rejection with this proposal needs fewer than two draws on average. At $\eta = 0$, reverse KL is infinite but rejection still costs exactly two draws on average. Odd parity is a low-target-probability event for $\eta > 0$, not literally outside the target support.

With access to exact conditional marginals, an alternative exact sampler uses just two rounds: draw $b-1$ uniform bits independently, then draw the last bit from its conditional distribution. Thus a failure of one-shot independent sampling is not a lower bound for adaptive sampling. All unconditional pairwise mutual informations vanish when $b \geq 3$. After sufficient reveals, dependencies become visible. This example warns against unqualified pairwise or marginal-entropy diagnostics; it does not contradict pairwise-to-joint theorems that assume additional structure.

J.2 Correlated bits: rejection cost is not adaptive depth

Let $p(0^b) = p(1^b) = 1/2$ and let $q = 2^{-b}$ be the product of its univariate marginals. Then

$$\mathrm{KL}(p\|q) = (b-1)\log 2, \qquad q(\mathrm{supp}(p)^c) = 1 - 2^{1-b}, \qquad \sup_{x:p(x)>0} p(x)/q(x) = 2^{b-1}. \tag{50}$$

The whole-block rejection bound is tight and exponential in b. Nevertheless, exact marginal-oracle sampling takes two rounds: sample one bit, then reveal all remaining bits, which are deterministic given the first. Large initial total correlation alone therefore does not establish large adaptive sampling depth.

J.3 What one set of marginal predictions cannot identify

Let p_+ be the noisy-even-parity law above and p_- its odd-parity counterpart. They give identical univariate marginals, but $\mathrm{TV}(p_+, p_-) = 1 - 2\eta$. Any procedure whose only target-dependent input is this one set of univariate marginals must return the same law r for both targets. The triangle inequality implies

$$\max\{\mathrm{TV}(r, p_+), \mathrm{TV}(r, p_-)\} \geq (1 - 2\eta)/2. \tag{51}$$

This is a restricted information-access statement. It excludes access to conditional queries after reveals, joint densities, learned joint heads, or target-specific side information. General adaptive-oracle lower bounds require more substantial arguments, as in Chen et al. (2025).

J.4 Event probabilities and the two KL directions

For any event A and laws p, q, data processing gives

$$d(q(A)\|p(A)) \leq \mathrm{KL}(q\|p), \tag{52}$$

$$d(p(A)\|q(A)) \leq \mathrm{KL}(p\|q), \tag{53}$$

where d is the binary KL. Pinsker bounds $|q(A) - p(A)|$ by the square root of either KL divided by two. Consequently, forward KL tending to zero cannot coexist with a fixed positive excess failure probability. Conversely, a large reverse KL does not certify a large failure probability for an arbitrarily chosen event. Metrics must be interpreted together with the event definition and target base rate.

If $p(A) = 0$ and $q(A) = \alpha$, an exact unchanged-proposal rejection rule must reject every sample in A; hence its acceptance rate is at most $1 - \alpha$ and its expected number of proposals is at least $1/(1 - \alpha)$. This remains a rejection-specific count, not a GPU latency or general correction bound.

J.5 Forward separation of model and dependence error

Fix a context and suppress it in the notation. For a joint law p_B, its marginals p_i, and learned marginals q_i,

$$\mathrm{KL}(p_B\|\textstyle\prod_i q_i) = \mathbb{E}_{p_B} \log \dfrac{p_B}{\prod_i p_i} + \displaystyle\sum_i \mathbb{E}_{p_B} \log \frac{p_i}{q_i} \tag{54}$$

$$= \mathrm{KL}(p_B\|\textstyle\prod_i p_i) + \sum_i \mathrm{KL}(p_i\|q_i). \tag{55}$$

This holds with the usual extended-real convention for missing support. Applying the KL chain rule across fixed blocks gives a target-prefix expectation of these terms. EB-Sampler and ParallelBench already analyze closely related decompositions (Ben-Hamu et al., 2025; Kang et al., 2025). Reversing every KL does not yield the same nonnegative additive identity.

J.6 Adaptive reveal orders

Let the state be the revealed set S and its values x_S. A deterministic policy $j = \Omega(S, x_S)$ chooses an unrevealed position. Use the same policy in target and model trajectories, changing only the conditional token kernels. A completed sequence uniquely determines its trajectory under this policy: replay the decisions from the empty state. With exact target conditionals, the resulting output law is p. The trajectory KL chain rule therefore yields

$$\mathrm{KL}(q_\Omega\|p) = \sum_{t=1}^{T} \mathbb{E}_{(S_t, x_{S_t}) \sim q_\Omega} \mathrm{KL}\Big(p_\theta(X_{\Omega(S_t, x_{S_t})} \mid x_{S_t}) \,\Big\|\, p(X_{\Omega(S_t, x_{S_t})} \mid x_{S_t})\Big). \tag{56}$$

The policy can use deterministic model scores, provided the identical state-to-position map is used in the target comparison. For a common randomized policy, retain the sampled actions in the path state. Policy likelihood ratios cancel in path KL, while output KL is bounded above by path KL via data processing; equality should not be asserted without a recoverability argument. These statements concern conditional sampling, not greedy argmax. If target-zero prefixes can be reached by the model, use extended-real KL and specify kernels at such prefixes; the finite bounded-error analysis instead requires the explicit uniform bound in Appendix A.

J.7 Adaptive block selection and joint error

Let $H_m = (S_m, x_{S_m})$ be the revealed state. A deterministic policy chooses a nonempty block $B_m = \Omega(H_m) \subseteq [T] \setminus S_m$ before sampling its new values. Use the same policy for the target and model trajectories. The target draws the exact joint $p(X_{B_m} \mid H_m)$; the factorized model draws $\prod_{i \in B_m} p_\theta(X_i \mid H_m)$. Each completed sequence determines its path under this policy, and the target output law is p by successive conditional sampling. Applying the chain rule and the fixed-context identity gives

$$\mathrm{KL}(p \| q_{\theta,\Omega}) = \mathbb{E}_{H_{1:M} \sim p,\Omega} \sum_{m=1}^{M} \Bigg[\mathrm{KL}\Bigg(p(X_{B_m} \mid H_m) \,\Bigg\|\, \prod_{i \in B_m} p(X_i \mid H_m) \Bigg) + \sum_{i \in B_m} \mathrm{KL}(p(X_i \mid H_m) \| p_\theta(X_i \mid H_m)) \Bigg]. \tag{57}$$

The number of rounds $M \leq T$ can depend on the revealed values; equivalently, pad terminated paths with empty updates of zero divergence. Both terms depend on the policy through its visited states and chosen blocks. Thus analytical separation of error sources does not imply that order and block size can be optimized independently.

The policy may use model confidence or entropy, provided it is a fixed function of the revealed state and is shared by the comparison trajectories. Selecting a block using newly sampled uncommitted values requires modeling that selection step too; the displayed expression must not be applied unchanged to such a procedure. For a common randomized policy, the same chain rule holds on paths including policy actions, with output KL bounded by path KL. This is an extension of the diagnostic bookkeeping, not a new optimal-schedule theorem.

J.8 A prefix mismatch certifies a local distribution change

Let u and v be categorical distributions before and after revealing a draft prefix. Suppose a is the unique maximizer of u, with margin $\Delta = u_a - \max_{j \neq a} u_j > 0$, and a different b is selected under v. Then $v_b \geq v_a$ and

$$\Delta \leq u_a - u_b = (u_a - v_a) + (v_a - v_b) + (v_b - u_b) \leq |u_a - v_a| + |u_b - v_b| \leq 2\,\mathrm{TV}(u, v). \tag{58}$$

Pinsker subsequently gives $\mathrm{KL}(u \| v) \geq \Delta^2/2$. This is a local certificate conditional on a changed maximizer. Agreement has no converse guarantee of distributional closeness. For learned masked models the change can reflect genuine dependence, conditional-model inconsistency, or both. Neither this certificate nor the observed mismatch rate estimates total correlation or global KL.