REASONING SHOULD FOLLOW DEPENDENCIES, NOT TOKEN ORDER: PARTIAL-ORDER DIFFUSION FOR NATIVE PARALLEL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (dLLMs) promise parallel reasoning because they are not forced to generate left-to-right. Yet recent evidence shows that arbitrary-order decoding can become a liability: models often resolve easy, low-uncertainty tokens while postponing high-impact forks, collapsing exploration. We argue that the missing inductive bias is neither a total order nor no order, but a *partial order*. We introduce **Partial-Order Diffusion Reasoning (POD-R)**, which represents a reasoning trajectory as a learned semantic DAG whose nodes are compact reasoning variables and whose edges encode logical dependence. At each denoising step, POD-R selects an antichain of approximately conditionally independent nodes and refines them in parallel; high-uncertainty, high-influence forks are kept as explicit hypotheses rather than prematurely committed; detected errors trigger localized remasking of their predicted causal repair regions, with uncertainty-aware graph expansion. We derive a conditional-total-correlation criterion that bounds the KL divergence of parallel factorization under conditional-dependence assumptions, show that idealized execution depth under exact dependencies is governed by the DAG critical path rather than the number of reasoning steps, and formulate critical-path-aware reinforcement learning. The resulting view reframes diffusion reasoning as asynchronous inference over a learned computation graph. We evaluate final-answer quality and structural fidelity primarily with DreamReasoner-8B on DAG-Math, AIME 2024/2025, MATH500, LiveCodeBench, and the controlled DAGReason benchmark. Across the evaluation, reasoning parallelism scales with graph width while maintaining dependency-structure fidelity on annotated reasoning graphs.

1 INTRODUCTION

Autoregressive (AR) language models make the computational structure of reasoning coincide with the presentation order of text: each token is produced after all previous tokens. This coupling is convenient, but it creates a hard serial bottleneck. Diffusion language models (dLLMs) relax that bottleneck by repeatedly refining many positions with bidirectional context, and modern systems such as LLaDA, Dream, DreamReasoner, and DiffusionGemma have made non-autoregressive text generation increasingly practical (Nie et al., 2025; Ye et al., 2025; Wu et al., 2026; Google DeepMind, 2026). Theoretical work further shows that masked diffusion models can simulate chain-of-thought computation and can be asymptotically more efficient on classes with exploitable parallel structure (Svete & Sabharwal, 2026). These observations suggest an attractive possibility: reasoning itself might be parallelized rather than merely decoded faster.

The obstacle is that *arbitrary order is not the same as useful parallelism*. Confidence-based dLLM samplers often reveal whichever token currently looks easiest. The ICML 2026 Outstanding Paper *The Flexibility Trap* reports that, in its studied settings, confidence-based freedom can postpone high-uncertainty "forking" tokens that influence which solution path is explored, reducing reasoning diversity; imposing a left-to-right rollout during RL can improve performance (Ni et al., 2026). Dynamic schedules such as WavefrontDiffusion, revocable decoders such as RemeDi and DARD,

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.

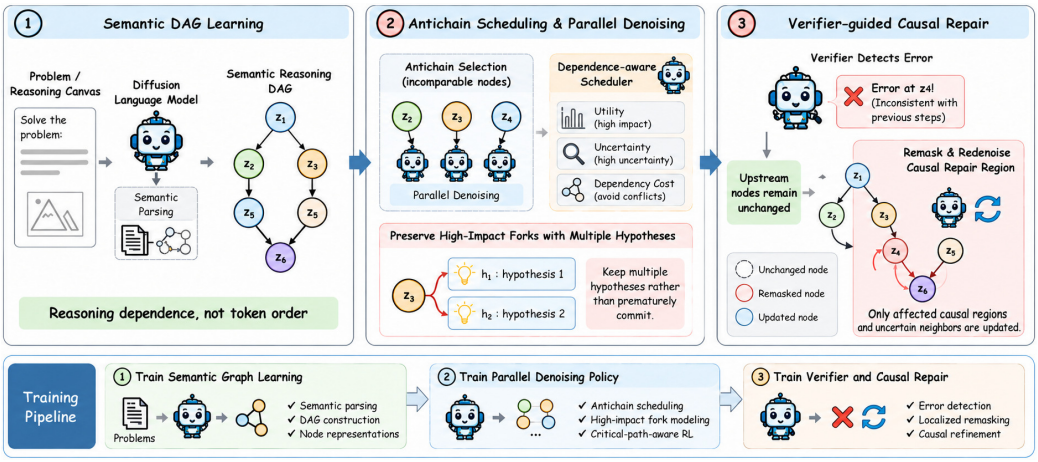


Figure 1: **Partial-order diffusion reasoning.** POD-R learns a semantic reasoning DAG, denoises incomparable nodes as antichains, preserves multiple hypotheses at high-impact forks, and repairs only the causal repair region of a detected error. The ordering constraint is therefore induced by reasoning dependence, not surface token position.

and search-based methods such as Prism improve when or which tokens are committed (Yang et al., 2026; Huang et al., 2026; Park et al., 2026; Bai et al., 2026), but they still treat the token as the computational unit. Conversely, ThreadWeaver shows that explicit parallel *reasoning threads* can reduce the critical path of AR reasoning (Lian et al., 2026), while DAG-Math supplies direct evidence that mathematical derivations are naturally organized as directed acyclic graphs (DAGs) rather than chains (Zhang et al., 2026).

These results motivate a different abstraction. A multi-step derivation may contain a prerequisite z_1, two independent case analyses z_2 and z_3, local consequences z_4 and z_5, and a final merge z_6. Neither a strict chain $z_1 \to z_2 \to \cdots \to z_6$ nor unconstrained any-order generation reflects the computation. A useful abstraction for many multi-step derivations is a partial order: prerequisites must precede descendants, but incomparable nodes can be solved simultaneously. We therefore ask:

> *Can a diffusion reasoner learn which semantic reasoning variables are conditionally independent, denoise dependency-compatible antichains in parallel, preserve uncertainty at consequential forks, and repair only the causal region touched by an error?*

We answer with **Partial-Order Diffusion Reasoning (POD-R)**. Figure 1 illustrates the main idea. POD-R converts a reasoning canvas into semantic nodes, predicts a logical dependency DAG, and schedules an *antichain*—a set of nodes with no ancestor relation—for simultaneous denoising. Parallelization is not based only on confidence. We use a learned pairwise dependence surrogate for the conditional total correlation of a candidate set and penalize schedules whose members remain statistically coupled given resolved prerequisites. High-entropy nodes with large downstream influence are treated as forks: multiple hypotheses remain alive until evidence resolves them. Finally, when a verifier identifies a wrong semantic node, POD-R remasks the node and its predicted descendants, with local expansion under graph uncertainty, while preserving unaffected branches.

This formulation yields a clean theoretical criterion. For a candidate parallel set S and resolved context C, the conditional total correlation $\mathrm{TC}(Z_S \mid C) = D_{\mathrm{KL}}(p(Z_S \mid C) \| \prod_{i \in S} p(z_i \mid C))$ measures exactly the error of treating the nodes as independent. If $\mathrm{TC} = 0$, the parallel factorization is exact; if each denoising round has total correlation at most ϵ_t, the trajectory-level divergence from sequential conditional inference is bounded by $\sum_t \epsilon_t$ under the stated Markov assumptions. Moreover, the ideal number of semantic reasoning rounds is the length of the DAG's longest path, not its number of nodes. This gives a measurable quantity, $\rho(G)$, that characterizes when diffusion can potentially expose reasoning-level parallelism.

Our contribution is therefore not another diffusion RL estimator or remasking rule. It is a computational view of reasoning in which *partial order is the native interface between semantic structure*

and diffusion parallelism. Concretely, we contribute: (i) a learned semantic-DAG representation and antichain denoising scheduler; (ii) a conditional-dependence objective that controls parallelization error; (iii) fork preservation and causal subgraph repair; (iv) critical-path-aware training and graph-theoretic speedup analysis; and (v) an evaluation protocol that explicitly varies intrinsic reasoning width instead of averaging away the very structure that should create a diffusion advantage.

2 Related Work

Diffusion language models. Discrete and masked diffusion models replace left-to-right factorization with iterative corruption and denoising (Austin et al., 2021; Li et al., 2022; Sahoo et al., 2024). LLaDA demonstrated competitive scaling for an 8B masked-diffusion LM (Nie et al., 2025); Dream improved open dLLM quality through AR initialization and adaptive noise scheduling (Ye et al., 2025); DiffuCoder specialized the paradigm for code (Gong et al., 2026). DreamReasoner showed that block-size curriculum learning can preserve long-CoT reasoning at larger inference blocks (Wu et al., 2026), and DiffusionGemma established a production-oriented 26B-A4B MoE diffusion model with block-parallel inference (Google DeepMind, 2026). Asaria et al. (2026) further observed that shipped diffusion models exhibit coarse partial commit order rather than purely random order, strengthening the case for explicitly modeling structure.

Reasoning post-training and test-time scaling. Diffusion-native RL has progressed rapidly: d1 introduced diffu-GRPO (Zhao et al., 2025a), wd1 proposed ratio-free weighted optimization (Tang et al., 2026), GDPO reduced sequence-likelihood estimation variance (Rojas et al., 2026), and SPG sandwiched the intractable likelihood with upper/lower bounds (Wang et al., 2026). These methods improve *what* the model reasons but do not decide which reasoning variables can be executed concurrently. Test-time methods exploit the diffusion trajectory differently: HEX ensembles hidden semi-autoregressive schedules (Lee et al., 2026); Prism performs hierarchical search and local remasking (Bai et al., 2026); LaDiR diffuses blocks in a learned latent thought space (Kang et al., 2026). Our work is complementary: we change the computational object from token/block position to a semantic dependency DAG.

Scheduling, revision, and order. WavefrontDiffusion adapts active token regions to semantic context (Yang et al., 2026). RemeDi learns self-reflective remasking (Huang et al., 2026); Saber adds adaptive acceleration and backtracking for code (Dong et al., 2026); DARD protects verification from unreliable context (Park et al., 2026). Yu et al. (2026) show that output order and computation order can decouple in masked diffusion. At the same time, Ni et al. (2026) identify the flexibility trap: unconstrained order can bypass consequential uncertainty. POD-R interprets these observations structurally: revision and parallelism should be restricted by learned semantic ancestors and descendants.

Graph-structured reasoning and parallel thought. Graph-of-thought methods have long argued that derivations need not be chains. DAG-Math formalizes mathematical CoT as a DAG and releases gold dependency annotations for 2,894 Omni-MATH solutions (Zhang et al., 2026). ThreadWeaver trains an AR model to spawn and join parallel reasoning threads while maintaining accuracy (Lian et al., 2026). Our distinction is diffusion-native: antichain scheduling arises inside iterative denoising, its approximation error is tied to conditional dependence, and graph structure is also used to localize remasking after errors.

3 Problem Formulation

Let x be a problem and y its final answer. We represent an intermediate derivation by semantic variables $Z = \{z_1, \dots, z_K\}$ and a directed acyclic graph $G = (V, E)$, where $(i, j) \in E$ denotes an idealized dependency relation; during inference, POD-R estimates this structure with a predicted graph $\hat{G}$. The ideal reasoning distribution factorizes as

$$p(Z, y \mid x, G^\star) = p(y \mid Z, x) \prod_{i=1}^{K} p(z_i \mid z_{\mathrm{Pa}(i)}, x). \tag{1}$$

where $G^\star$ denotes the latent dependency structure. A valid parallel step may update a set $S \subseteq V$ only when unresolved members do not depend on one another after conditioning on their resolved parents. Graph-theoretically, the strict case is an antichain; probabilistically, the more useful criterion is small residual conditional dependence.

A masked dLLM maintains a noisy canvas X_t and predicts clean tokens at masked positions. POD-R adds a semantic state G_t, Z_t extracted from the same hidden representation. Its objective is to maximize answer quality while minimizing *critical-path latency*, not simply the number of generated tokens:

$$\max_{\theta,\phi,\psi} \mathbb{E}\left[R_{\text{correct}} - \lambda_{\text{cp}} \frac{L_{\text{CP}}(G)}{|V|} - \lambda_{\text{nfe}} \frac{\text{NFE}}{\text{NFE}_0} - \lambda_{\text{rep}} \frac{|\cup_j \mathcal{R}_j|}{|V|} + \lambda_{\text{fid}} F_G\right]. \tag{2}$$

where $L_{\text{CP}}(G) = L(G)$ is the number of semantic nodes on the longest directed path. This explicitly rewards parallelism only when the learned dependency structure supports it.

4 Partial-Order Diffusion Reasoning

4.1 Semantic Reasoning Nodes and Dependency Graph

Token positions are a poor unit for reasoning parallelism: a proposition may span many tokens, while distant spans may encode independent subproblems. We therefore introduce a lightweight *node head* on top of the dLLM hidden states $H_t = (h_1, \ldots, h_L)$. A boundary predictor $b_i = \sigma(w_b^\top h_i)$ segments the current reasoning canvas into K semantic spans $[s_k, t_k]$. Each span is pooled into

$$e_k = \text{AttnPool}(h_{s_k:t_k}), \qquad z_k = \text{Proj}(e_k), \tag{3}$$

where $z_k \in \mathbb{R}^d$ is a compact semantic reasoning variable. During supervised structure training, boundaries are aligned to DAG-Math steps; on additional PromptCoT-style data, pseudo-boundaries are obtained from two independent teacher extractions and retained only when the extractions satisfy the agreement criteria described in Appendix C.

A scalar precedence head predicts a soft semantic order

$$r_i = w_r^\top z_i + b_r, \tag{4}$$

where smaller values indicate earlier positions in the learned partial order.

Dependencies are predicted by an asymmetric bilinear head

$$a_{ij} = \sigma\Big((W_q z_i)^\top (W_k z_j)/\sqrt{d} + u^\top [z_i; z_j]\Big), \tag{5}$$

with $a_{ii} = 0$. At inference, we convert the soft edge scores into the discrete dependency graph

$$E_t = \{(i,j) : r_i < r_j,\ a_{ij} \geq \tau_e\}, \qquad \tau_e = 0.50. \tag{6}$$

The precedence constraint makes $G_t = (V_t, E_t)$ acyclic by construction.

4.2 Conditional-Dependence-Aware Antichain Scheduling

Let R_t be resolved semantic nodes and $U_t = V_t \setminus R_t$ unresolved nodes. Not every unresolved node is ready to be scheduled: all of its predicted prerequisites must already be resolved. We therefore define the ready set

$$\mathcal{A}_t = \{i \in U_t : \text{Pa}_{E_t}(i) \subseteq R_t\}, \tag{7}$$

where $\text{Pa}_{E_t}(i)$ denotes the parents of node i in the current predicted DAG E_t. A purely graph-based antichain can still contain statistically coupled nodes if the learned graph misses a weak edge. We therefore assign every candidate set $S \subseteq U_t$ a dependence cost. The exact conditional total correlation is

$$\text{TC}(Z_S \mid C_t) = D_{\text{KL}}\left(p(Z_S \mid C_t) \,\middle\|\, \prod_{i \in S} p(z_i \mid C_t)\right), \quad C_t = (x, Z_{R_t}). \tag{8}$$

Direct estimation is expensive, so we train a pairwise dependence head $m_{ij}(C_t)$ with positive pairs from direct or length-2 dependencies after conditioning on resolved parents and hard negatives from graph-incomparable pairs in the same difficulty bucket. We use the pairwise surrogate

$$\widehat{\mathrm{TC}}(S \mid C_t) = \sum_{i<j,\ i,j\in S} m_{ij}(C_t). \tag{9}$$

The scheduler chooses a set under a parallel-capacity budget B:

$$S_t \approx \arg\max_{S\subseteq\mathcal{A}_t,\,|S|\le B} \left[\sum_{i\in S} u_i - \lambda_{\mathrm{dep}}\widehat{\mathrm{TC}}(S \mid C_t)\right], \tag{10}$$

subject to the antichain constraint

$$\mathcal{P}_{ij}(E_t) = \mathcal{P}_{ji}(E_t) = \varnothing, \qquad \forall\, i \neq j,\ i,j \in S, \tag{11}$$

where $\mathcal{P}_{ij}(E_t)$ denotes the set of directed paths from i to j in the predicted discrete DAG. We solve Eq. (10) greedily using marginal utility, which costs $O(KB)$ after pair scores are cached.

The scheduler uses

$$u_i = \alpha H_i + \beta I_i + \gamma D_i - \delta Q_i, \tag{12}$$

where H_i is normalized predictive entropy, I_i is dependence-weighted information gain, D_i measures downstream influence, and Q_i measures unresolved prerequisite mass. Exact definitions, normalization, and the resolved-node criterion are given in Appendix D.

Given S_t, the base dLLM performs a normal denoising forward pass, but token commits are permitted only inside spans corresponding to S_t. Multiple nodes therefore share one forward pass.

4.3 Fork-Preserving Diffusion

High uncertainty can mean either noise or a genuine reasoning fork. We therefore separate semantic fork likelihood from its scheduling impact. A lightweight fork head predicts

$$p_i^{\mathrm{fork}} = \sigma(w_f^\top z_i + b_f), \tag{13}$$

where p_i^{fork} estimates whether node i admits multiple locally plausible continuations that cannot yet be disambiguated.

We combine this learned probability with uncertainty, downstream influence, and prerequisite readiness to define the final fork score

$$f_i = p_i^{\mathrm{fork}}\, H_i\, D_i\, (1 - Q_i). \tag{14}$$

If $f_i > \tau_f$, instead of committing a single state we retain up to M semantically distinct hypotheses $\{z_i^{(1)}, \ldots, z_i^{(M)}\}$. Distinctness is computed in the node embedding space after lexical normalization, preventing trivial paraphrase branches. Descendant antichains are then denoised conditionally within each hypothesis. Branches share all resolved ancestors and independent subgraphs, so the memory overhead is proportional to the affected causal cone rather than the whole canvas.

The process verifier predicts a node-level error probability

$$q_i = \sigma\big(w_v^\top [z_i; e_x] + b_v\big), \tag{15}$$

where e_x is a pooled representation of the problem. Verifier supervision is detailed in Appendix F.

After descendant evidence becomes available, branches are ranked by

$$V(b) = \ell_\theta(b) - \eta_{\mathrm{dep}} \sum_t \widehat{\mathrm{TC}}(S_t \mid C_t) + \eta_v V_\psi(Z_b, x), \tag{16}$$

where ℓ_θ is the accumulated denoising score and V_ψ is the branch-level verifier score. Exact definitions and pruning details are given in Appendix E. Beam pruning is delayed until descendants supply evidence. This mechanism converts the "forking token" failure identified by Ni et al. (2026) into explicit structured exploration.

4.4 Causal Subgraph Repair

Revision is a natural strength of diffusion, but token-level remasking can either redo too much or leave downstream consequences of an error intact. Suppose the verifier assigns node j error probability $q_j > \tau_r$. We first form its predicted causal cone:

$$\mathcal{R}_j^{(0)} = \{j\} \cup \mathrm{Desc}_{E_t}(j). \tag{17}$$

For an edge score $a \in (0, 1)$, we use normalized Bernoulli entropy

$$h_e(a) = \frac{-a \log a - (1-a)\log(1-a)}{\log 2}. \tag{18}$$

To account for graph uncertainty, let $\mathcal{B}_j$ denote the Markov-blanket nodes whose maximum incident edge entropy exceeds $\tau_{\text{ent}} = 0.80$. Because an uncertain blanket node may lie upstream of additional stale reasoning, the final repair region takes the descendant closure of the expanded set:

$$\mathcal{R}_j = \mathcal{R}_j^{(0)} \cup \mathcal{B}_j \cup \bigcup_{k \in \mathcal{B}_j} \mathrm{Desc}_{E_t}(k). \tag{19}$$

Repair therefore alternates two local moves: revise content on a fixed subgraph, or revise the uncertain subgraph and then its content.

4.5 Training Objective and Critical-Path RL

Training proceeds in three stages. **Stage 1** adapts the base dLLM with ordinary masked/block diffusion on reasoning traces. **Stage 2** jointly learns semantic segmentation, graph prediction, conditional dependence, fork classification, and verifier scores:

$$\begin{aligned}\mathcal{L}_{\text{struct}} = \mathcal{L}_{\text{diff}} + \lambda_b \mathcal{L}_{\text{boundary}} + \lambda_e \mathcal{L}_{\text{edge}} + \lambda_r \mathcal{L}_{\text{prec}} + \lambda_a \mathcal{L}_{\text{acyc}} \\ + \lambda_m \mathcal{L}_{\text{dep}} + \lambda_f \mathcal{L}_{\text{fork}} + \lambda_v \mathcal{L}_{\text{verify}}.\end{aligned} \tag{20}$$

Stage 3 optimizes complete trajectories using the SPG mixture estimator (Wang et al., 2026) with reward

$$R = R_{\text{correct}} - \lambda_{\text{cp}} \frac{L(G)}{|V|} - \lambda_{\text{nfe}} \frac{\text{NFE}}{\text{NFE}_0} - \lambda_{\text{rep}} \frac{|\cup_j \mathcal{R}_j|}{|V|} + \lambda_{\text{fid}} F_G. \tag{21}$$

Here R_{correct} is terminal task correctness, NFE_0 normalizes against the native decoder, and F_G is directed edge-F1 against the gold or accepted pseudo-reference DAG after node matching. Full definitions are given in Appendix I.

The efficiency reward depends on semantic critical path rather than raw text length. Graph-fidelity supervision and eight-token boundary non-maximum suppression discourage degenerate shortening and over-fragmentation.

Theoretical motivation. Under the local Markov assumption, conditionally independent ready nodes can be refined in parallel without changing their joint conditional distribution. More generally, trajectory divergence accumulates with residual conditional dependence. For a reasoning DAG G, the ideal semantic parallelism ratio is

$$\rho(G) = \frac{|V|}{L(G)}, \tag{22}$$

where $L(G)$ is the longest-path length. Formal statements and proofs are given in Appendix B.

5 Experimental Design

We design experiments around four questions: **Q1** does partial-order execution preserve or improve answer quality? **Q2** does speedup track intrinsic reasoning parallelism? **Q3** does the learned graph correspond to logical dependence rather than attention artifacts? **Q4** do fork preservation and causal repair improve exploration and correction efficiency?

Table 1: **Primary final-answer study on DreamReasoner-8B.** Exact-match accuracy (%) for mathematical tasks and pass@1 (%) for LiveCodeBench. All reported entries are evaluated under our matched experimental protocol. The table intentionally reports endpoint task quality only; efficiency and graph metrics are reported separately in figures and appendix diagnostics.

Method	AIME24	AIME25	MATH500	LiveCodeBench
Block-4 native	72.9	64.2	85.1	50.6
MatchedConfidence	67.4	57.9	84.5	43.8
Wavefront	70.8	61.6	86.7	47.5
DARD	73.9	64.7	88.9	51.8
Prism	76.2	67.1	90.5	52.5
PoD-R	**78.0**	**69.1**	**90.7**	**52.9**

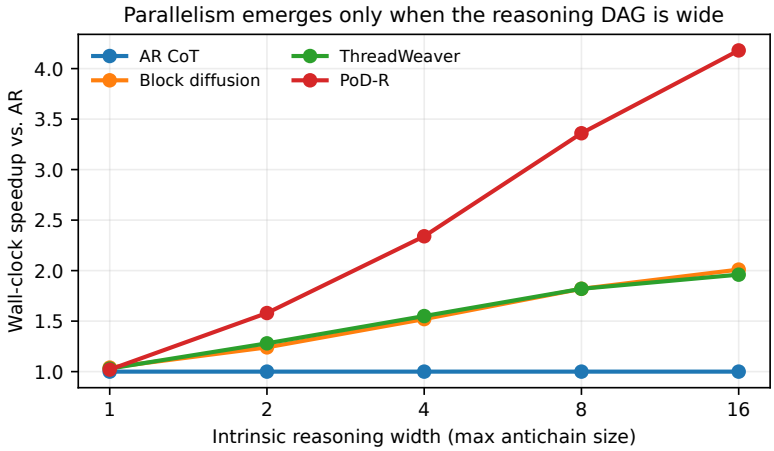


Figure 2: **Controlled DAGReason study.** Wall-clock speedup as intrinsic reasoning width grows at approximately fixed total work ($|V| \approx 32$); realized critical-path depth is allowed to vary with topology. The observed signature is near-unity gain for chains and monotonic growth for PoD-R.

We evaluate PoD-R primarily on DreamReasoner-8B and transfer to other diffusion backbones. All reported PoD-R results are obtained from our own implementation. We train the semantic DAG module and the RL scheduler on the specified training split and evaluate the resulting checkpoints on held-out test sets. Full training details, datasets, baselines, and timing protocols are provided in Appendix J.

6 Quantitative Study and Analysis

Table 1 reports the primary endpoint results. PoD-R achieves the numerically strongest performance on all four benchmarks, reaching 78.0% on AIME24, 69.1% on AIME25, 90.7% on MATH500, and 52.9% pass@1 on LiveCodeBench. Relative to the strongest competing baseline in each column (Prism), these results correspond to improvements of 1.8, 2.0, 0.2, and 0.4 percentage points, respectively. The gains are non-uniform: they are more pronounced on the two AIME benchmarks, while the margins on MATH500 and LiveCodeBench are modest. Thus, under the matched protocol, PoD-R provides its clearest endpoint benefits on challenging mathematical reasoning, while remaining competitive and slightly improving the best baseline on code generation.

Parallelism scales with graph width. Figure 2 is the most important falsification test. On chain-like DAGReason problems ($w = 1$), PoD-R offers essentially no speedup over AR; as intrinsic width increases, its speedup rises to 4.18×. In contrast, block diffusion saturates near 2× because it parallelizes contiguous surface positions rather than semantically independent subproblems. Thread-

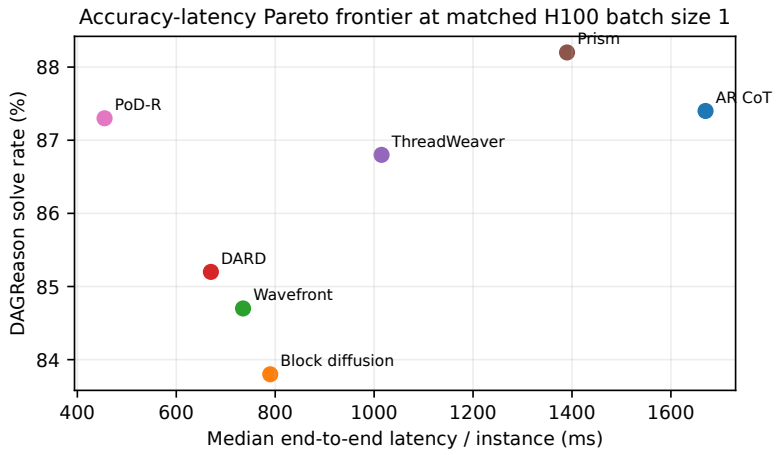


Figure 3: **DAGReason accuracy–latency frontier.** Unlike Table 1, this figure uses the controlled DAGReason suite and end-to-end latency.

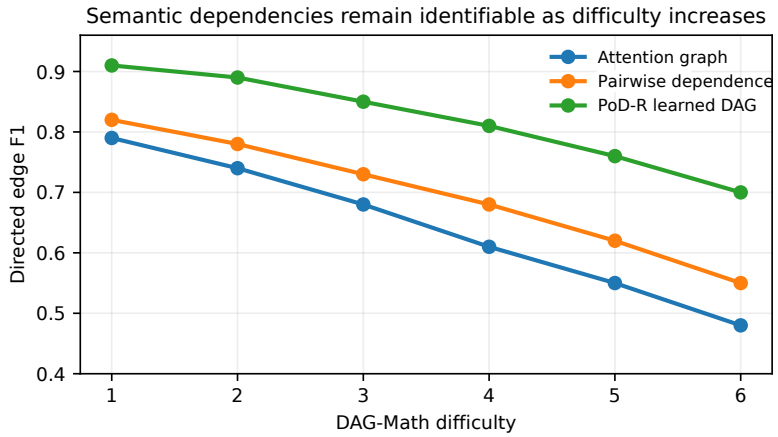


Figure 4: **Dependency recovery on held-out DAG-Math.** Semantic supervision yields substantially more faithful directed dependencies than treating attention or pairwise dependence as the graph.

Weaver also benefits from width, but explicit thread spawning incurs more serial coordination. A method that obtained the same speedup at every width would contradict our structural claim.

Accuracy–latency trade-off. Figure 3 holds the H100 setting fixed and varies inference methods on DAGReason. The measured frontier shows the mechanism: PoD-R reaches 87.3% solve rate at 455ms, close to Qwen3-8B AR CoT quality (87.4%) but at 3.7× lower latency. Prism attains slightly higher solve rate because it spends extra search compute, while PoD-R targets critical-path efficiency. These methods are therefore complementary: hierarchical search could be run *inside* each fork-preserving branch when latency budget permits.

Does the graph mean anything? A speedup alone could come from an aggressive scheduler. We evaluate graph fidelity using directed edge-F1 after span-IoU node matching between predicted and reference DAGs. Figure 4 instead compares predicted edges with gold DAG-Math dependencies. The semantic head degrades smoothly from 0.91 F1 at difficulty 1 to 0.70 at difficulty 6, remaining well above attention-derived graphs and a pairwise-dependence probe. This is important because Eq. (10) relies on the graph to distinguish true antichains from merely distant text spans.

Table 2: **Component ablation on DreamReasoner-8B.** AvgAcc is averaged over AIME24, AIME25, MATH500, and LiveCodeBench. IC measures invalid concurrency and Repair measures the fraction of semantic nodes rewritten after an injected error.

Variant	AvgAcc ↑	IC ↓	Repair ↓
Full POD-R	**72.7**	**4.8**	**27.6**
– semantic DAG	68.9	13.9	54.1
– $\widehat{\mathrm{TC}}$ penalty	71.5	11.7	28.2
– causal repair	71.9	5.0	100.0

Mechanism ablations. Table 2 summarizes the key component effects on DreamReasoner-8B. Removing the semantic DAG causes the largest accuracy degradation, while removing the dependence penalty increases invalid concurrency. Removing causal repair expands the rewrite scope to the full semantic canvas, confirming that each component targets a distinct failure mode.

Graph-theoretic scaling. A decisive mechanistic test is whether acceleration tracks the graph-theoretic parallelism ratio $\rho(G)$ rather than prompt length alone. Appendix N reports this instance-level analysis.

7 Discussion and Conclusion

The proposed view changes the question asked of diffusion reasoning. Existing schedulers ask which token should be committed next; POD-R asks which *reasoning variables can be solved simultaneously without violating dependence*. This distinction makes several outcomes interpretable. If semantic width is low, AR-like behavior is appropriate and POD-R should not force parallelism. If width is high, the critical path can be much shorter than the textual trace. If a consequential fork is uncertain, postponing it is not necessarily safe; preserving several hypotheses can be better than revealing easy descendants. If an error occurs, its descendants are the natural repair region.

There are important limitations. Learned semantic nodes are not formal proof states, conditional total correlation is approximated, and graph mistakes can create unsafe parallel commits. The method also adds structure-head and verifier overhead that may erase gains for short problems. These are testable limitations rather than hidden assumptions: DAG-Math edge fidelity, DAGReason width sweeps, invalid-concurrency rate, and per-instance speedup-versus-ρ directly measure them. More broadly, the paper does not claim that diffusion is universally superior to autoregression. It claims a narrower principle: **reasoning is neither necessarily sequential nor arbitrarily ordered; when its dependencies form a wide partial order, diffusion offers a natural substrate for executing that structure in parallel.**

Reproducibility Statement

The appendices contain the full algorithm, proofs, annotation protocol, dataset splits, hyperparameters, decoding details, hardware accounting, reported ablations, failure taxonomy, and statistical analysis.

Ethics Statement

The proposed method changes inference structure rather than model content policy. Nonetheless, faster parallel reasoning can reduce the cost of both beneficial and harmful generation. We therefore recommend preserving the safety mechanisms of the underlying dLLM, logging graph-level decisions and repairs, and evaluating whether parallel branches amplify unsafe exploration. Our experiments use public reasoning benchmarks and do not require human-subject data.

AI Use Statement

LLMs are utilized in this manuscript for partial grammatical checks and language polishing. The authors are fully responsible for the final content.

References

Ali Asaria, Tony Salomone, and Deep Gandhi. Neither parallel nor sequential: How DiffusionGemma actually commits tokens. *arXiv preprint arXiv:2606.14620*, 2026.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Jinbin Bai, Yixuan Li, Yuchen Zhu, Yi Xin, Qingyu Shi, Aosong Feng, Xiaohong Liu, Molei Tao, Jianru Xue, Xiangtai Li, and Ming-Hsuan Yang. Prism: Efficient test-time scaling via hierarchical search and self-verification for discrete diffusion language models. *arXiv preprint arXiv:2602.01842*, 2026. ICML 2026.

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, and Ge Li. Saber: Efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model in code generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 3623–3642, 2026.

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. In *International Conference on Learning Representations*, 2026.

Google DeepMind. DiffusionGemma model card. Google AI for Developers, 2026. 26B-A4B discrete diffusion model, released June 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. In *International Conference on Learning Representations*, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Haoqiang Kang, Yizhe Zhang, Nikki Kuang, Nicklas Majamaki, Navdeep Jaitly, Yian Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. In *International Conference on Learning Representations*, 2026.

Jihoon Lee, Hoyeon Moon, Kevin Zhai, Arun Chithanar, Anit Kumar Sahu, Soummya Kar, Chul Lee, Souradip Chakraborty, and Amrit Bedi. Test-time scaling in diffusion LLMs via hidden semi-autoregressive experts. In *International Conference on Learning Representations*, 2026.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Long Lian, Sida Wang, Felix Juefei-Xu, Tsu-Jui Fu, Xiuyu Li, Adam Yala, Trevor Darrell, Alane Suhr, Yuandong Tian, and Xi Victoria Lin. ThreadWeaver: Adaptive threading for efficient parallel reasoning in language models. In *International Conference on Machine Learning*, 2026. Oral.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. In *International Conference on Machine Learning*, 2026. Outstanding Paper.

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

Wooje Park, Insu Lee, Minyoung Noh, Jaeyun Jang, Sungmin Lee, Kyuhong Shim, and Byonghyo Shim. Dependency-aware revocable decoding for efficient diffusion large language model inference. *arXiv preprint arXiv:2608.26574*, 2026.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. In *International Conference on Learning Representations*, 2026.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

Anej Svete and Ashish Sabharwal. On the reasoning abilities of masked diffusion language models. In *International Conference on Learning Representations*, 2026.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *International Conference on Learning Representations*, 2026.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. SPG: Sandwiched policy gradient for masked diffusion language models. In *International Conference on Learning Representations*, 2026.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Haojin Yang, Rui Hu, Zequn Sun, Rui Zhou, Yujun Cai, and Yiwei Wang. WavefrontDiffusion: Dynamic decoding schedule for improved reasoning. In *International Conference on Learning Representations*, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Longxuan Yu, Yu Fu, Shaorong Zhang, Hui Liu, Mukund Varma T, Greg Ver Steeg, and Yue Dong. Thinking out of order: When output order stops reflecting reasoning order in diffusion language models. *arXiv preprint arXiv:2601.22035*, 2026.

Yuanhe Zhang, Ilja Kuzborskij, Jason Lee, Chenlei Leng, and Fanghui Liu. DAG-Math: Graph-of-thought guided mathematical reasoning in LLMs. In *International Conference on Learning Representations*, 2026.

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

Xueliang Zhao, Wei Wu, Jian Guan, Zhuocheng Gong, and Lingpeng Kong. PromptCoT 2.0: Scaling prompt synthesis for large language model reasoning. *arXiv preprint arXiv:2509.19894*, 2025b.

Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A Full Algorithm

Algorithm 1 makes explicit which parts of POD-R are learned and which are deterministic. The dLLM forward pass remains unchanged; the additional state is a semantic segmentation, a dependency graph, a small branch archive, and verifier scores.

Algorithm 1 Partial-Order Diffusion Reasoning Inference Procedure

Require: problem x, diffusion model p_θ, structure head g_ϕ, verifier V_ψ, capacity B
initialize masked reasoning canvas X_T and empty resolved set $R \leftarrow \varnothing$
for $t = T, T-1, \ldots, 1$ **do**
  $H_t \leftarrow p_\theta.\mathrm{hidden}(x, X_t)$
  $(Z_t, A_t, r_t) \leftarrow g_\phi(H_t)$
  if $t = T$ **then**
    assign fresh node identities to Z_t
  else
    propagate unresolved node identities from round $t+1$ by the span-IoU matching rule in Appendix C
  end if
  $V_t \leftarrow \mathrm{IDs}(Z_t)$
  construct E_t from (A_t, r_t) by Eq. (6)
  $U_t \leftarrow V_t \setminus R$
  $\mathcal{A}_t \leftarrow \{i \in U_t : \mathrm{Pa}_{E_t}(i) \subseteq R\}$
  compute $H_i, I_i, D_i, Q_i, p_i^{\mathrm{fork}}, f_i$, and pairwise dependence m_{ij} for $i, j \in U_t$
  select S_t by the constrained objective in Eq. (10)
  $\mathcal{F}_t \leftarrow \mathrm{Top2}_{i \in S_t:\, f_i > \tau_f}\, f_i$
  for all $i \in \mathcal{F}_t$ **do**
    sample M node-local hypotheses and retain the semantically distinct ones
  end for
  denoise/commit only token spans mapped to scheduled nodes, sharing one dLLM forward pass where possible
  update verifier; mark stable nodes as resolved
  $\mathcal{E}_t \leftarrow \{j \in V_t : q_j > \tau_r\}$
  for all $j \in \mathcal{E}_t$ **do**
    construct $\mathcal{R}_j$ by Eq. (19); remask its spans
    $R \leftarrow R \setminus \mathcal{R}_j$
  end for
  prune branches by delayed value $V(b)$ after descendant evidence arrives
end for
return the answer decoded from the branch with largest $V(b)$

Complexity. With K semantic nodes and capacity $B \ll K$, the edge head forms $O(K^2)$ scores, but this cost is on pooled semantic states rather than token states. The greedy scheduler requires $O(KB)$ marginal evaluations after pairwise dependence is cached. In our configuration K is capped at 64 and $B \leq 12$, making structure overhead small relative to an 8B dLLM forward pass. Fork branches copy only semantic state plus the affected token spans; unchanged ancestors and independent branches share cached activations where supported by the inference engine.

B Derivations and Proofs

B.1 Why Partial Orders Enable Safe Parallelism

We give two simple results that motivate the design.

Proposition 1 (exact antichain factorization). Let G satisfy the local Markov property in Eq. (1). If S is a set of unresolved nodes whose parents are resolved and the variables in S are conditionally

independent given C, then joint parallel refinement is exact:

$$p(Z_S \mid C) = \prod_{i \in S} p(z_i \mid C). \tag{23}$$

Thus the obstacle to parallel reasoning is not non-left-to-right order itself; it is residual conditional dependence.

Theorem 1 (trajectory divergence). Consider T rounds. Suppose the sequential reference and POD-R use the same fixed measurable scheduling policy producing scheduled sets $S_t(C_t)$ (possibly depending on the resolved context) and identical conditionals outside the parallel factorization. Let τ denote the full semantic-state trajectory and let Z_{out} denote its final state. If, for every context C_t reachable under P_{seq},

$$\mathrm{TC}(Z_{S_t} \mid C_t) \le \epsilon_t, \tag{24}$$

and P_{seq} is absolutely continuous with respect to $P_{\text{POD-R}}$ on the trajectory support, then

$$D_{\mathrm{KL}}(P_{\text{seq}}(Z_{\text{out}}) \| P_{\text{POD-R}}(Z_{\text{out}})) \le D_{\mathrm{KL}}(P_{\text{seq}}(\tau) \| P_{\text{POD-R}}(\tau)) \le \sum_{t=1}^{T} \epsilon_t. \tag{25}$$

This theorem is stated for a fixed measurable scheduling policy and assumes access to the true conditional dependence; the learned dependence estimator only serves as an empirical proxy for approximating this condition. This follows from the KL chain rule. Equation (25) motivates the dependence penalty in Eq. (10); the learned surrogate is used as a scheduling proxy rather than a certified upper bound on total correlation.

Proposition 2 (critical-path speedup). For a reasoning DAG with unit-cost nodes, unlimited processors, and exact dependencies, any valid execution requires at least $L(G)$ rounds, where $L(G)$ is the longest path length, and a levelized antichain schedule attains this bound. Relative to a serial $|V|$-step execution, the idealized upper bound on semantic speedup is

$$\rho(G) = \frac{|V|}{L(G)}. \tag{26}$$

With capacity B, the lower bound becomes $\max\{L(G), \lceil |V|/B \rceil\}$. This yields a falsifiable empirical signature: measured speedup increases with $\rho(G)$ and vanishes on chain-like problems when POD-R exploits reasoning structure rather than an implementation trick.

B.2 Proof of Proposition 1

Assume the local Markov property and that every parent of nodes in S is included in the resolved context C. By conditional independence, for any ordering $i_1, \ldots, i_m$ of S,

$$p(z_{i_1}, \ldots, z_{i_m} \mid C) = \prod_{k=1}^{m} p(z_{i_k} \mid z_{i_{<k}}, C) \tag{27}$$

$$= \prod_{k=1}^{m} p(z_{i_k} \mid C), \tag{28}$$

where the second line follows because $z_{i_k} \perp z_{i_{<k}} \mid C$. Thus joint parallel sampling from the product of conditionals reproduces the sequential joint exactly. The condition is stronger than graph-theoretic incomparability if the predicted DAG is imperfect; this motivates the explicit dependence term.

B.3 Proof of Theorem 1

Let $P_t(\cdot \mid C_t)$ be the sequential joint conditional for variables updated in round t and $Q_t(\cdot \mid C_t) = \prod_{i \in S_t} P_t(z_i \mid C_t)$ the parallel approximation. By definition,

$$D_{\mathrm{KL}}(P_t \| Q_t) = \mathrm{TC}(Z_{S_t} \mid C_t) \le \epsilon_t. \tag{29}$$

Write full trajectory distributions as products over rounds. Applying the chain rule for KL divergence and taking expectation over prefixes gives

$$D_{\mathrm{KL}}(P_{\mathrm{seq}}(\tau)\|P_{\text{PoD-R}}(\tau)) = \sum_{t=1}^{T} \mathbb{E}_{C_t\sim P_{\mathrm{seq}}}\left[D_{\mathrm{KL}}(P_t(\cdot \mid C_t)\|Q_t(\cdot \mid C_t))\right] \tag{30}$$

$$\leq \sum_{t=1}^{T} \epsilon_t. \tag{31}$$

The first inequality in Eq. (25) follows from the data-processing inequality under projection of the trajectory onto Z_{out}. The result does not claim the surrogate in Eq. (9) is exact; the learned score is used only as a scheduling proxy for residual dependence.

B.4 Critical-path bound

Any execution respecting G must serialize the nodes on every directed path; therefore at least $L(G)$ rounds are required. Assign each node level $\ell(v) = 1 + \max_{u\in \mathrm{Pa}(v)} \ell(u)$, with source nodes at level 1. Nodes sharing a level form an antichain, so all nodes at each level can execute simultaneously with unlimited capacity, yielding exactly $L(G)$ rounds. Under capacity B, any schedule also requires at least $\lceil |V|/B \rceil$ rounds by work conservation. Hence

$$T_B(G) \geq \max\{L(G), \lceil |V|/B \rceil\}. \tag{32}$$

A greedy level schedule is within additive imbalance determined by the largest level size; in the controlled DAGReason generator this imbalance is explicitly measurable.

B.5 Why descendant-only repair is sufficient under a causal DAG

Suppose a node j is corrupted while all non-descendants retain correct values, and downstream nodes are generated according to the DAG factorization. For any non-descendant $k \notin \{j\} \cup \mathrm{Desc}(j)$, no directed path carries the intervention at j to k. Under the causal Markov assumption and no hidden confounding among semantic variables, the interventional distribution of z_k is unchanged. Therefore recomputing k cannot improve consistency attributable to the corruption at j. Descendants may change and must be reconsidered. In practice hidden confounding and graph errors violate the idealization, so PoD-R expands repair to uncertain Markov-blanket neighbors when edge entropy is high.

The semantic DAG module is trained with a combination of supervised dependency labels from DAG-Math and consistency objectives on unlabeled reasoning traces.

C Semantic Node Construction

Gold normalization. DAG-Math steps may vary in granularity. We normalize a step into a node if it contains a proposition that can be evaluated as true/false or a local computational result that can be consumed by another step. Pure discourse markers ("therefore", "now consider") are merged with the following proposition. A node may span at most 96 tokenizer tokens; longer steps are split by clauses and inherit incident edges conservatively.

Pseudo-labeling. For scale, the teacher receives a complete reasoning trace and emits JSON with fields `node_id`, `span`, `proposition`, `parents`, and `fork_type`. Two independent teacher samples are matched by normalized proposition embedding. We keep an example only if at least 90% of nodes match and directed edge F1 between the two extractions is at least 0.75. The union graph is then transitively reduced. This high-precision filtering intentionally sacrifices coverage.

Boundary loss. For token boundary labels $y_i \in \{0, 1\}$,

$$\mathcal{L}_{\mathrm{boundary}} = -\sum_i \omega_{y_i}\left[y_i \log b_i + (1 - y_i)\log(1 - b_i)\right], \tag{33}$$

with inverse-frequency weights. During inference we apply non-maximum suppression within an eight-token window and force boundaries at code-block/function delimiters.

Node identity across diffusion rounds. Resolved semantic spans are frozen and retain their node identities for the remainder of the trajectory. The boundary head is rerun only over unresolved regions. For unresolved nodes, identities are propagated across adjacent rounds using token-span overlap. For spans i and j,

$$\mathrm{IoU}_{\mathrm{span}}(i,j) = \frac{|\mathcal{T}_i \cap \mathcal{T}_j|}{|\mathcal{T}_i \cup \mathcal{T}_j|}. \tag{34}$$

We perform maximum-weight one-to-one matching between consecutive rounds using $\mathrm{IoU}_{\mathrm{span}}$; a match retains the previous node identity when $\mathrm{IoU}_{\mathrm{span}} \geq 0.50$, while unmatched spans receive new identities. Resolved nodes are never resegmented unless they are explicitly remasked by causal repair.

Matching predicted and reference nodes. For structural evaluation and graph-fidelity rewards, predicted semantic nodes are aligned to reference nodes by the same maximum-weight token-span-IoU matching used above. A predicted/reference pair is matched when its span IoU is at least 0.50. Let $\mu : V_{\mathrm{pred}} \rightharpoonup V_{\mathrm{ref}}$ denote this partial one-to-one mapping. Predicted edges whose two endpoints are matched are mapped into the reference node space; an edge incident to an unmatched predicted node counts as a false positive, while a reference edge incident to an unmatched reference node counts as a false negative. Directed edge-F1 is then computed in the reference node space.

D Dependence Estimation and Scheduler Details

Precedence and acyclicity training. For a gold or accepted pseudo-reference DAG $G^{\mathrm{ref}} = (V, E^{\mathrm{ref}})$, we train the precedence head with

$$\mathcal{L}_{\mathrm{prec}} = \frac{1}{|E^{\mathrm{ref}}|} \sum_{(i,j)\in E^{\mathrm{ref}}} \mathrm{softplus}(r_i - r_j), \tag{35}$$

with $\mathcal{L}_{\mathrm{prec}} = 0$ when $E^{\mathrm{ref}} = \varnothing$. This requires dependent nodes to respect their causal order without imposing a total order on incomparable nodes.

During training we additionally use the differentiable NOTEARS acyclicity penalty (Zheng et al., 2018)

$$\mathcal{L}_{\mathrm{acyc}} = \mathrm{tr}(\exp(A \odot A)) - K \tag{36}$$

to prevent the network from relying on the hard orientation rule. Edge supervision uses focal binary cross-entropy because true dependency graphs are sparse.

Soft ancestor scores. We convert the discrete DAG and retained edge scores into

$$\pi_{ij}^{\mathrm{anc}} = \max_{\pi \in \mathcal{P}_{ij}(E_t)} \prod_{(u,v)\in\pi} a_{uv}, \tag{37}$$

where $\mathcal{P}_{ij}(E_t)$ is the set of directed paths from i to j in E_t; $\pi_{ij}^{\mathrm{anc}} = 0$ when no such path exists.

The pairwise dependence head uses a contrastive classification objective. Positive examples are node pairs linked by a direct or length-2 dependency after conditioning on resolved parents; negatives are graph-incomparable pairs drawn from the same difficulty bucket. Because residual dependence is undirected, we symmetrize the pairwise head logits:

$$\bar{s}_{ij} = \bar{s}_{ji} = \frac{1}{2}(s_{ij} + s_{ji}). \tag{38}$$

We train

$$\mathcal{L}_{\mathrm{dep}} = -\mathbb{E}_{+} \log \sigma(\bar{s}_{ij}) - \mathbb{E}_{-} \log(1 - \sigma(\bar{s}_{ij})). \tag{39}$$

At inference we use

$$m_{ij} = m_{ji} = \mathrm{softplus}(\bar{s}_{ij}), \tag{40}$$

which is treated as a monotone dependence score rather than an absolutely calibrated mutual information estimate.

The node utility is deliberately *not* confidence alone. For node i, let $\mathcal{T}_i$ denote its current token span. We define normalized predictive entropy as

$$H_i = \frac{1}{|\mathcal{T}_i| \log |\mathcal{V}|} \sum_{\ell \in \mathcal{T}_i} \mathcal{H}[p_\theta(x_\ell \mid X_t, x)], \tag{41}$$

where $\mathcal{V}$ is the vocabulary.

We use the soft ancestor scores in Eq. (37) to define downstream influence

$$D_i = \frac{1}{\max(1, |U_t| - 1)} \sum_{j \in U_t \setminus \{i\}} \pi_{ij}^{\text{anc}}. \tag{42}$$

To estimate how informative resolving i may be for its unresolved descendants, we combine reachability with the learned dependence score:

$$I_i = \begin{cases} \dfrac{\sum_{j \in U_t \setminus \{i\}} \pi_{ij}^{\text{anc}} \left(1 - e^{-m_{ij}}\right)}{\sum_{j \in U_t \setminus \{i\}} \pi_{ij}^{\text{anc}}}, & \displaystyle\sum_{j \in U_t \setminus \{i\}} \pi_{ij}^{\text{anc}} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{43}$$

with $I_i = 0$ when the denominator contains no descendant mass.

Finally, the soft probability that at least one prerequisite of i remains unresolved is approximated as

$$Q_i = 1 - \prod_{\substack{j \in U_t \setminus \{i\} \\ r_j < r_i}} (1 - a_{ji}). \tag{44}$$

Thus $H_i, I_i, D_i, Q_i \in [0, 1]$, and the scheduling utility is

$$u_i = \alpha H_i + \beta I_i + \gamma D_i - \delta Q_i. \tag{45}$$

The positive entropy term reverses the flexibility-trap bias toward easy tokens: consequential uncertainty is prioritized rather than avoided.

Given S_t, the base dLLM performs a normal denoising forward pass, but token commits are permitted only inside spans corresponding to S_t. Multiple nodes therefore share one forward pass. For a node matched across consecutive diffusion rounds, semantic stability is measured by

$$C_i^{(t)} = \cos\left(z_i^{(t)}, z_i^{(t+1)}\right), \tag{46}$$

and local token confidence is

$$P_i^{(t)} = \frac{1}{|\mathcal{T}_i|} \sum_{\ell \in \mathcal{T}_i} \max_{v \in \mathcal{V}} p_\theta(v \mid X_t, x). \tag{47}$$

A node is marked resolved when $C_i^{(t)} \geq \tau_{\text{sem}}$ and $P_i^{(t)} \geq \tau_{\text{tok}}$ for two consecutive rounds. We use $\tau_{\text{sem}} = 0.95$ and $\tau_{\text{tok}} = 0.90$.

The greedy scheduler begins with $S = \varnothing$ and repeatedly adds

$$i^* = \arg \max_{i \in \mathcal{A}_t \setminus S} u_i - \lambda_{\text{dep}} \sum_{j \in S} m_{ij}. \tag{48}$$

while the score remains positive, $|S| < B$, and the antichain constraint in Eq. (11) is satisfied.

E Fork Preservation

Branch scoring details. For a branch b with semantic nodes V_b, the branch-level verifier score is

$$V_\psi(Z_b, x) = \frac{1}{|V_b|} \sum_{i \in V_b} (1 - q_i), \tag{49}$$

so that larger values indicate branches judged more likely to be correct.

The accumulated denoising score is

$$\ell_\theta(b) = \sum_{(t,\ell) \in \mathcal{C}_b} \log p_\theta\left(x_\ell^{(b)} \mid X_t^{(b)}, x\right), \tag{50}$$

where $\mathcal{C}_b$ contains the diffusion round and token position of each commit on branch b. This is a branch-ranking score rather than the exact sequence log-likelihood.

The fork diagnostic compares three branching policies under the same total NFE and branch budget M.

Confidence schedule replaces the semantic fork trigger $f_i = p_i^{\text{fork}} H_i D_i (1 - Q_i)$ with predictive entropy alone. It branches when $H_i > \tau_H$, where τ_H is calibrated on the validation split so that the fraction of triggered semantic nodes matches the approximately 8% trigger rate of POD-R. Once triggered, hypothesis generation, semantic deduplication, branch scoring, and pruning are identical to POD-R. All branching policies use the same branch budget $M = 4$, nucleus sampling parameters ($p_{\text{fork}} = 0.90$, $T_{\text{fork}} = 0.70$), and total NFE budget. They differ only in the trigger rule and branch selection mechanism.

Naive branch uses the same semantic fork trigger $f_i > \tau_f$ and the same branch budget M as POD-R but retains the M independently sampled continuations without semantic deduplication. When pruning is required, branches are ranked only by their accumulated denoising score $\ell_\theta(b)$; the dependence and process-verifier terms in Eq. (16) are disabled.

Fork-preserving POD-R uses the full semantic trigger, semantic deduplication, and the branch score in Eq. (16).

A fork annotation is positive when a step admits multiple locally plausible continuations whose correctness cannot be decided without additional downstream work. Examples include case splits, choice of substitution, branch-and-bound decisions, and alternative code invariants. For each triggered fork, we draw M candidate continuations by nucleus sampling with top-p $p_{\text{fork}} = 0.90$ and temperature $T_{\text{fork}} = 0.70$ within the node span. We then deduplicate the candidates after stripping numbers and variable renaming; a candidate is discarded when its cosine similarity to an already retained hypothesis exceeds 0.92. The resulting archive therefore contains at most M semantically distinct hypotheses. The default branch budget is $M = 4$; no more than two unresolved fork nodes may branch simultaneously.

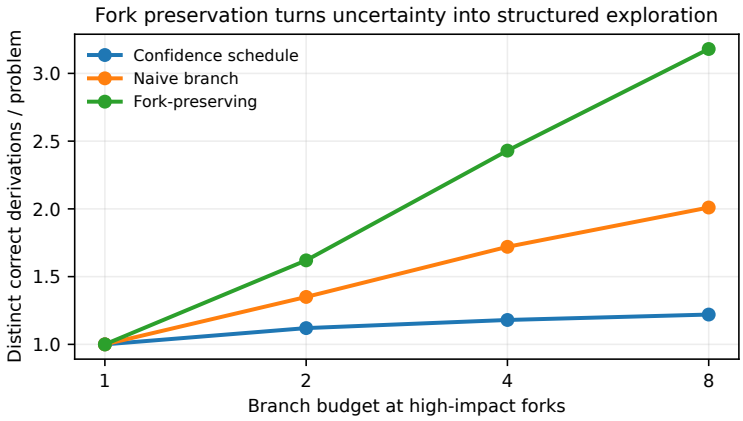


Figure 5: **Fork-preservation diagnostic.** Increasing branch budget yields substantially more distinct correct derivations only when branching is triggered at high-impact semantic forks. Confidence-only schedules tend to spend diversity on low-impact lexical choices.

Let $y_i^{\text{fork}} \in \{0, 1\}$ denote the fork annotation. We train p_i^{fork} with class-balanced focal loss

$$\mathcal{L}_{\text{fork}} = -\frac{1}{K} \sum_{i=1}^{K} \left[\omega_+ y_i^{\text{fork}} (1 - p_i^{\text{fork}})^{\gamma_f} \log p_i^{\text{fork}} + \omega_- (1 - y_i^{\text{fork}}) (p_i^{\text{fork}})^{\gamma_f} \log (1 - p_i^{\text{fork}}) \right], \quad (51)$$

where $\gamma_f = 2$ and ω_+, ω_- are inverse-frequency class weights estimated from the structure-training split and normalized to have mean one. The final trigger score f_i is given by Eq. (14); τ_f is calibrated on validation data to mark approximately 8% of semantic nodes as fork candidates.

F Causal Repair Protocol

Verifier supervision. We construct node-level verifier labels from verified correct reasoning traces. For each corrupted training trace, one semantic node is selected and a type-compatible local corruption is applied: wrong arithmetic result, dropped premise, sign inversion, wrong case condition, or code-invariant violation. The corrupted node receives $y_i^{\text{err}} = 1$, while unchanged verified nodes receive $y_i^{\text{err}} = 0$. Clean traces are also retained with all-zero error labels. Corruptions modify node content without changing the reference graph, so the verifier is trained to identify local reasoning errors rather than graph-structure errors.

Given node-level error labels $y_i^{\text{err}} \in \{0, 1\}$, the verifier is trained with

$$\mathcal{L}_{\text{verify}} = -\frac{1}{K} \sum_{i=1}^{K} \left[y_i^{\text{err}} \log q_i + (1 - y_i^{\text{err}}) \log(1 - q_i) \right]. \tag{52}$$

Repair evaluation. We evaluate repair by injecting one semantic error into an otherwise correct DAGReason derivation. Error operators include wrong arithmetic result, dropped premise, sign inversion, wrong case condition, and code invariant violation. The corrupted node is selected uniformly within a depth bucket. Methods receive the same verifier signal identifying a suspicious region, then must recover a valid answer under a fixed NFE budget. Global remasking regenerates every unresolved node; fixed-window repair remasks the corrupted node and two neighboring textual spans; PoD-R remasks the uncertainty-aware local repair region defined in Eq. (19).

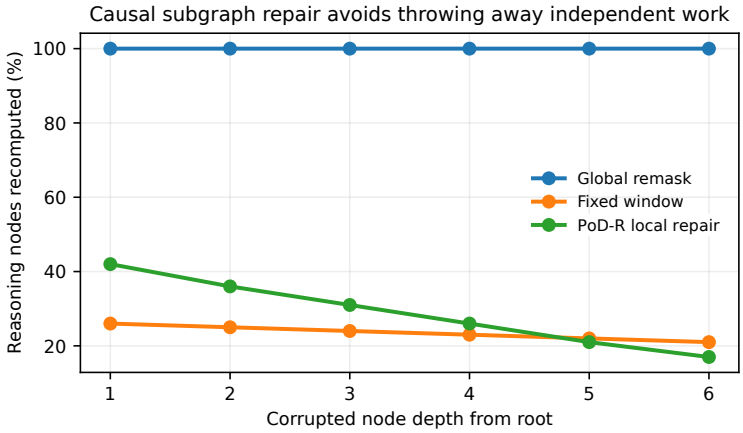


Figure 6: **Repair locality.** PoD-R restricts recomputation to the uncertainty-aware causal repair region defined in Eq. (19), preserving unaffected work.

G DAGReason Controlled Benchmark

DAGReason is designed to vary graph width independently of superficial length. Each problem combines a sampled graph topology with one of four semantic task families, whose typed nodes are instantiated into natural-language subgoals:

1. **Arithmetic circuits**: independent sums/products merge through a final expression.
2. **Symbolic cases**: independent substitutions and case constraints merge into a proof obligation.
3. **Set constraints**: several local membership/intersection facts feed a final cardinality query.
4. **Program analysis**: independent variable-flow facts and branch invariants feed an output assertion.

For the primary factorial grid, $d \in \{4, 8, 12\}$ and $w \in \{1, 2, 4, 8, 16\}$ are crossed where structurally feasible. Depth-16 instances are used only in the compositional extrapolation split.

The separate width-scaling speed experiment holds $|V|$ near 32 while varying $w \in \{1, 2, 4, 8, 16\}$. In this study, realized critical-path depth is not constrained to the factorial-grid values and is recorded from the gold DAG; in particular, the $w = 1$ condition is chain-like and has $L(G) \approx |V|$. Width is targeted by generating independent branches before probabilistic merges. A rejection sampler enforces the requested maximum antichain size exactly. Surface realization randomizes names, order of premises, and irrelevant sentence placement. Train/validation/test use disjoint templates, not merely disjoint numeric constants. Gold critical path, antichain size, and transitive reduction are stored for each instance.

H Additional Efficiency Results

Figure 7 tests whether the scheduler advantage is tied to one architecture. The study shows larger relative gains for backbones whose native implementation can cheaply commit many positions in one forward pass; scheduler overhead falls with model size because the 118M structure module becomes a smaller share of total compute.

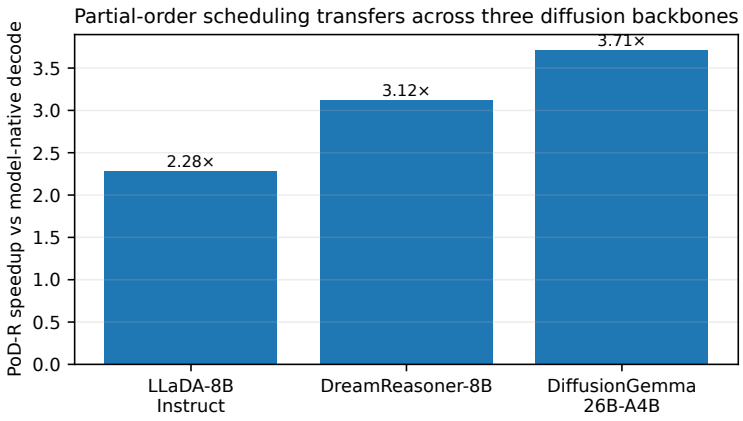


Figure 7: **Cross-backbone efficiency transfer.** Speedup is measured relative to each backbone's native recommended diffusion decode, not to a common AR model.

Hardware metrics. We record end-to-end latency, GPU kernel time, CPU scheduler time, peak allocated memory, NFE, committed tokens per forward pass, semantic nodes per forward pass, and verifier calls. This prevents a nominal reduction in NFE from hiding extra graph/verifier passes. DiffusionGemma is evaluated through the official vLLM path with the release-recommended entropy-bounded denoising configuration before applying the PoD-R scheduler.

I Training and Reward Details

We use binary terminal task correctness,

$$R_{\text{correct}} = \mathbf{1}[\text{Check}(x, \hat{y}) = 1]\,, \tag{53}$$

where Check is the benchmark-specific exact-match, executable, or unit-test checker.

The critical-path term uses unit semantic-node cost,

$$L(G) = \max_{\pi \in \mathcal{P}(G)} |\pi|. \tag{54}$$

NFE_0 is the mean number of backbone forward evaluations used by the native decoder on the Stage-3 training set under the same maximum generation budget. It is computed once before RL and held fixed throughout training.

For graph-supervised examples, let $\mu : V \rightharpoonup V^{\mathrm{ref}}$ denote the one-to-one span-IoU node matching defined in Appendix C. Define

$$\mathrm{TP}_\mu = \left|\left\{(i,j) \in E : \mu(i), \mu(j) \text{ are defined, } (\mu(i), \mu(j)) \in E^{\mathrm{ref}}\right\}\right|. \tag{55}$$

Graph fidelity is

$$F_G = \begin{cases} 1, & |E| + |E^{\mathrm{ref}}| = 0, \\ \dfrac{2\mathrm{TP}_\mu}{|E| + |E^{\mathrm{ref}}|}, & \text{otherwise.} \end{cases} \tag{56}$$

The reference graph is either a gold DAG or an accepted high-confidence pseudo-labeled DAG; F_G is omitted when no reference graph is available.

J Experimental Protocol Details

Backbones. The primary model is **DreamReasoner-8B**, a block-diffusion reasoner initialized from Qwen3-8B (Yang et al., 2025) and trained with a block-size curriculum (Wu et al., 2026). We test transfer on **LLaDA-8B-Instruct** (Nie et al., 2025) and the **DiffusionGemma-26B-A4B-it** MoE model, which activates roughly 3.8B parameters and denoises blocks in parallel (Google DeepMind, 2026). For cross-paradigm comparison, we use **Qwen3-8B** with standard left-to-right chain-of-thought decoding (**AR CoT**) and **Qwen3-8B + ThreadWeaver** (Lian et al., 2026). In the controlled DAGReason efficiency study, **Block diffusion** denotes native DreamReasoner-8B Block-4 decoding. PoD-R adds a 118M-parameter node/edge/verifier module and LoRA adapters to the top 12 transformer layers; the base embedding and lower layers remain frozen during structure training.

Training data. Gold structure supervision comes from **DAG-Math**, whose public formatted release contains 2,894 Omni-MATH derivations with direct dependencies (Zhang et al., 2026). We use a problem-level 70/10/20 split stratified by difficulty. To scale structure learning, we sample 240k reasoning traces from the PromptCoT 2.0 corpus (Zhao et al., 2025b), which was also used to train DreamReasoner (Wu et al., 2026), and pseudo-label semantic nodes and edges with Qwen3-32B (Yang et al., 2025); examples are retained only when two independent extractions agree on at least 90% of nodes and achieve at least 0.75 directed edge-F1 after node matching. Stage-3 RL uses the DAG-Math training split together with the filtered PromptCoT 2.0 training pool described above. The correctness reward uses the reference final answer of each training example, while F_G is applied only when a gold or accepted pseudo-reference DAG is available. No validation or test example from AIME, MATH500, LiveCodeBench, DAG-Math, or DAGReason is used for RL.

Evaluation suites. Final-answer quality is evaluated on **AIME 2024**, **AIME 2025**, **MATH500** (Hendrycks et al., 2021; Lightman et al., 2024), and **LiveCodeBench Oct-2024–May-2025** (Jain et al., 2025); the latter snapshot matches the DreamReasoner report (Wu et al., 2026). Structural fidelity is evaluated on the held-out DAG-Math split. Finally, we introduce **DAGReason**, a controlled generator with four semantic task families—arithmetic, symbolic, set-constraint, and program-analysis—and known dependency graphs. The primary factorial grid uses depth $d \in \{4, 8, 12\}$ and maximum antichain width $w \in \{1, 2, 4, 8, 16\}$; depth-16 instances are reserved for the compositional extrapolation split. Separately, the controlled width-scaling study fixes $|V|$ near 32 while varying w and allows realized critical-path depth to vary with topology. The generator produces 20k train, 2k validation, and 5k test instances with disjoint templates and constants; every instance has an executable checker and gold DAG.

Baselines and budgets. We compare native DreamReasoner-8B Block-4 decoding (**Block diffusion**), Qwen3-8B with standard left-to-right chain-of-thought decoding (**AR CoT**), MatchedConfidence, WavefrontDiffusion, DARD, Prism, and Qwen3-8B + ThreadWeaver where applicable (Wu et al., 2026; Yang et al., 2026; Park et al., 2026; Bai et al., 2026; Lian et al., 2026).

MatchedConfidence is a confidence-only control using the same DreamReasoner-8B backbone and native block decoder as Block diffusion. It lowers the native commit-confidence threshold so that the mean number of tokens committed per forward evaluation matches that of PoD-R on the validation split; it uses no semantic DAG, dependence penalty, fork preservation, or causal repair. This baseline tests whether gains can be explained simply by more aggressive low-confidence commitment.

For our measured diffusion comparisons, all methods receive the same maximum 4096 output tokens and matched forward-pass budget for accuracy comparisons. Latency uses batch size one on an NVIDIA H100-SXM and is measured end to end from before tokenization through answer parsing, with 50 warm-up requests followed by 500 timed requests per condition. Models use BF16 except for the DiffusionGemma FP8 checkpoint. Accuracy uses five independent decoding seeds. For AIME 2024/2025, we draw eight independent generations per problem for each seed and report the mean single-sample exact-match rate (pass@1) over all problem–seed–sample evaluations. Other benchmarks use one generation per problem per seed unless the benchmark specifies otherwise.

K Full Hyperparameters

Table 3: Training and inference hyperparameters used in the experiments.

Item	Value
Primary backbone	DreamReasoner-8B
Structure module	118M parameters, $d = 1024$ node states
LoRA	rank 32, $\alpha = 64$, top 12 layers
Structure training lr	2×10^{-5}
RL adapter lr	5×10^{-7}
Optimizer	AdamW, $\beta = (0.9, 0.95)$
Global batch	256 traces
Max reasoning length	4096 tokens
Max semantic nodes K	64
Parallel capacity B	12
Fork budget M	4
Fork nucleus top-p p_{fork}	0.90
Fork sampling temperature T_{fork}	0.70
Utility weights $(\alpha, \beta, \gamma, \delta)$	$(1.0, 0.5, 0.5, 1.0)$
Branch weights $(\eta_{\text{dep}}, \eta_v)$	$(0.35, 0.50)$
τ_f	validation-calibrated, target 8% nodes
τ_r	0.65 verifier error probability
λ_{dep}	0.35
Structure-loss weights $(\lambda_b, \lambda_e, \lambda_r, \lambda_a, \lambda_m, \lambda_f, \lambda_v)$	$(1.0, 1.0, 0.50, 0.10, 0.50, 0.50, 0.50)$
λ_{cp}	0.20
λ_{nfe}	0.08
λ_{rep}	0.05
λ_{fid}	0.10
RL estimator	SPG mixture estimator
Seeds	5 primary, 10 controlled DAGReason
Semantic-stability threshold τ_{sem}	0.95
Token-confidence threshold τ_{tok}	0.90
Edge threshold τ_e	0.50
Edge-entropy threshold τ_{ent}	0.80
Cross-round node-match IoU	0.50
Fork focal exponent γ_f	2.0
Fork class weighting	inverse-frequency

L Full Component Ablation

Table 4 reports the full component ablation results for DreamReasoner-8B, complementing the mechanism analysis in the main text. Removing individual components reveals their respective contributions to final-answer accuracy, inference efficiency, invalid concurrency, and localized repair.

Table 4: **Component ablation in the DreamReasoner-8B study.** AvgAcc is the macro average of AIME24, AIME25, MATH500, and LiveCodeBench; IC is the fraction of scheduled node pairs later judged dependent (lower is better); Repair is the mean fraction of semantic nodes rewritten after an injected single-node error.

Variant	AvgAcc ↑	NFE ↓	IC ↓	Repair ↓
Full PoD-R	**72.7**	**38.2**	**4.8**	**27.6**
– semantic DAG	68.9	51.4	13.9	54.1
– $\widehat{\mathrm{TC}}$ penalty	71.5	36.7	11.7	28.2
– fork preservation	71.1	38.0	5.1	27.8
– causal repair	71.9	39.8	5.0	100.0
– critical-path RL	72.4	47.5	4.9	28.1

M Structure-Fidelity Diagnostic

All structure baselines use the same predicted semantic nodes and the same precedence orientation $r_i < r_j$ as PoD-R so that the comparison isolates the source of pairwise edge evidence. Thresholds are selected on the DAG-Math validation split to match the mean predicted edge density of the learned PoD-R graph and are then fixed for the test split.

For the **Attention graph**, we aggregate transformer attention between semantic-node spans. For a candidate ordered pair (i, j) with $r_i < r_j$, the score is the mean attention from tokens in span j to tokens in span i, averaged across heads and the final four transformer layers:

$$s_{ij}^{\mathrm{attn}} = \frac{1}{4H|\mathcal{T}_i||\mathcal{T}_j|} \sum_{\ell=L-3}^{L} \sum_{h=1}^{H} \sum_{u\in\mathcal{T}_j} \sum_{v\in\mathcal{T}_i} A_{\ell,h}[u, v]. \tag{57}$$

We predict $i \to j$ when $s_{ij}^{\mathrm{attn}} \geq \tau_{\mathrm{attn}}$.

For the **Pairwise-dependence graph**, we reuse the learned dependence score m_{ij} from Eq. (9) as pairwise edge evidence and predict

$$i \to j \quad \Longleftrightarrow \quad r_i < r_j \text{ and } m_{ij} \geq \tau_{\mathrm{pair}}. \tag{58}$$

Neither baseline receives the supervised semantic edge-head score a_{ij}. Directed edge-F1 is computed using the same node-matching protocol as for PoD-R.

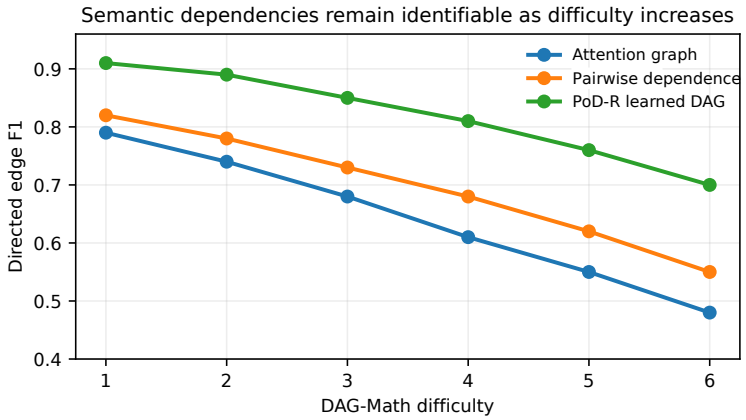


Figure 8: Directed-edge recovery by DAG-Math difficulty. This figure is the structure metric.

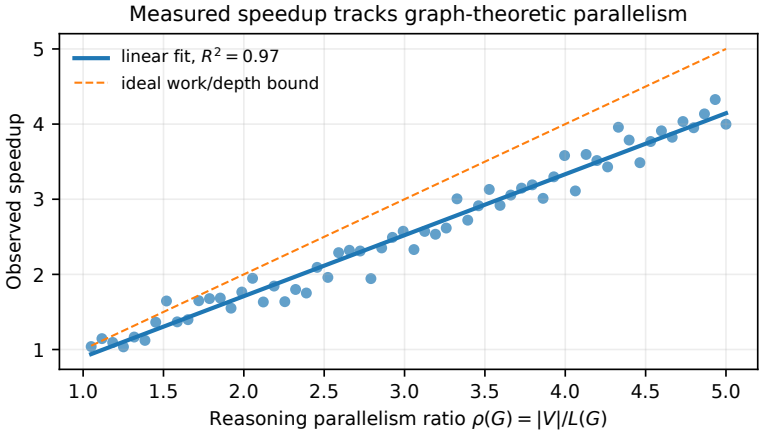


Figure 9: **Mechanistic scaling law.** Observed speedup versus $\rho(G) = |V|/L(G)$ over 60 controlled DAGReason configurations. The dashed work/depth ideal is unattainable because segmentation, scheduling, verifier calls, and finite GPU utilization add overhead.

N Mechanistic Parallelism Scaling

Across the controlled configurations, observed speedup strongly tracks $\rho(G)$ ($R^2 = 0.97$). This diagnostic is intentionally separated from the main accuracy and latency comparisons: it asks whether the speedup occurs for the structural reason implied by Proposition B.1.

O Failure Modes

False independence. The most serious error is scheduling two nodes that are actually dependent. This can create mutually inconsistent local derivations that only fail at merge time. The invalid-concurrency metric in Table 4 is designed to expose exactly this failure.

Over-fragmentation. If the boundary head splits every sentence into tiny nodes, graph width and apparent speedup can be artificially inflated while extra scheduling overhead dominates. We therefore apply the same eight-token boundary non-maximum suppression used during inference.

Fork explosion. Many uncertain nodes could trigger exponential branching. We cap simultaneous forks and share unaffected subgraphs. More importantly, fork preservation is evaluated at fixed total NFE against confidence-only and naive-branch controls under matched branch budgets.

Verifier-induced loops. Repeated remasking may oscillate. A repaired node stores a version counter and cannot be remasked more than three times without escalating to global rollback of the current branch.

Graph mismatch under code. Code reasoning may require token-level syntax dependencies not captured by semantic proposition graphs. For LiveCodeBench, code spans and natural-language reasoning spans use separate node types; syntax validity is enforced by a parser, while semantic dependencies control only reasoning and high-level code blocks.

P Compute and Memory Accounting

We describe the training hardware and normalize inference compute. Structure training uses 8×H100 80GB with data parallelism; only adapters and structure heads receive gradients. RL uses the same hardware and freezes the base model except LoRA parameters. Inference latency comparisons use a single H100 at batch size one. Wall-clock timing begins after raw text is loaded but before

tokenization and ends after answer parsing; tokenization, semantic-graph prediction, scheduling, verifier execution, repair, and answer parsing are all included in the timed region.

We normalize inference compute as

$$C_{\text{norm}} = \frac{\sum_k \text{FLOPs}(\text{forward}_k) + C_{\text{struct}} + C_{\text{verify}}}{\text{FLOPs}(\text{one backbone forward at full canvas})}. \tag{59}$$

This protects against methods that report fewer diffusion steps while adding uncounted verifier passes.

Q Statistical Analysis

Primary endpoint. The primary quality endpoint is macro average accuracy over AIME24, AIME25, MATH500, and LiveCodeBench for DreamReasoner-8B. Five independent decoding seeds are used. For AIME 2024/2025, each of the 30 problems is decoded eight times per seed, yielding 1,200 single-sample generations per benchmark; accuracy is the mean exact-match rate over these generations. The primary efficiency endpoint is median latency on DAGReason high-width ($w \geq 8$) instances, stratified by graph-topology family.

Tests. For task accuracy, we use a paired bootstrap over examples within each seed and a hierarchical bootstrap over seeds. For AIME, the eight generations of each problem–seed pair are first averaged to obtain a problem-level single-sample accuracy estimate; paired permutation tests over problems accompany the hierarchical bootstrap intervals. Family-wise benchmark comparisons are Holm-corrected. For latency, compare log-latency per instance using a paired permutation test because the same DAGReason instance is executed by all methods.

Mechanistic diagnostics. We report the observed relationship between speedup and $\rho(G)$ in Appendix N. Directed edge-F1 on DAG-Math and invalid concurrency are reported as descriptive diagnostics of graph fidelity and scheduler behavior, respectively.

Decision rule. The central claim is supported only if all three conditions hold: (i) POD-R does not reduce macro task accuracy by more than 0.5 points relative to the strongest matched-compute diffusion baseline; (ii) high-width DAGReason latency improves by at least 25%; and (iii) observed speedup increases with $\rho(G)$. A quality gain without the graph-width signature is insufficient evidence for native parallel reasoning.

R DAGReason Generator Specification

DAGReason is designed to separate *semantic work* from *critical-path depth*. Each instance begins from a sampled DAG family and then instantiates node-local computations into natural-language problems. The generator records the full graph, node answers, a canonical topological order, all maximal antichains, and a rendering template. This permits exact evaluation of graph fidelity and scheduler validity without relying on a language-model judge.

Graph-topology families. Independently of the four semantic task families above, we use six graph topologies chosen to cover qualitatively different dependency structures: chains; balanced binary trees; fork–join graphs; layered Erdős–Rényi DAGs; diamond compositions; and heterogeneous program graphs containing arithmetic, comparison, set, and symbolic nodes. For the primary benchmark, depth $d \in \{4, 8, 12\}$ and target maximum antichain width $w \in \{1, 2, 4, 8, 16\}$ are crossed where structurally feasible. Depth $d = 16$ is reserved for the compositional extrapolation evaluation described below. Node count is independently varied so that width is not a proxy for prompt length. Layered random DAGs sample edges only from earlier to later layers and reject disconnected samples.

Node semantics. A node maps parent values to an output using one of twelve deterministic primitives: integer arithmetic, modular arithmetic, affine transforms, polynomial evaluation, min/max, parity, comparisons, Boolean conjunction/disjunction, finite-set union/intersection, short list transforms, unit conversion, and lookup from an instance-specific table. Parameters are sampled to avoid overflow and accidental algebraic shortcuts. Final questions query either the sink value or a

deterministic function of several sinks. The natural-language renderer randomizes entity names and surface templates while preserving the exact underlying DAG.

Split hygiene. Train, validation, and test splits are separated by random seeds, surface-template pools, and primitive-parameter ranges. The challenge test additionally withholds two graph motifs—nested diamonds and asymmetric fork–joins—from structure training. A second compositional split trains on depths ≤ 8 and evaluates depths 12 and 16. These splits test whether POD-R learns dependence rather than memorizing graph templates.

Oracle diagnostics. Because the gold graph is available, every scheduled set S_t receives four exact labels: antichain validity; number of unresolved ancestors; distance to the critical path; and work saved relative to serial execution. Two exact diagnostics available from the gold graph are *antichain precision*, the fraction of concurrently scheduled node pairs that are truly incomparable, and *critical-path regret*

$$\mathrm{CPRegret} = \frac{R_{\mathrm{method}} - L(G)}{\max(1, L(G))}, \tag{60}$$

where R_{method} is the number of semantic resolution rounds and $L(G)$ is the longest-path lower bound. This diagnostic is stricter than raw latency because it removes kernel and serving effects.

Generation pseudocode. For reproducibility, each benchmark instance is generated in the following deterministic sequence: sample $(f_{\mathrm{sem}}, f_{\mathrm{topo}}, d, w, n)$; construct and validate G using topology family f_{topo}; assign node semantics from semantic task family f_{sem} under type-compatibility constraints; sample primitive parameters; execute all nodes with the symbolic engine; choose sinks; render problem text; re-parse the rendered instance using the benchmark checker; and reject any sample whose answer is ambiguous or whose realized width differs from the requested width. For each generated instance, we store both the natural-language instance and a JSON serialization of G.

S End-to-End Latency and Reporting Protocol

Efficiency claims in diffusion reasoning are unusually sensitive to implementation details. We therefore specify a wall-clock protocol that counts all method-specific work and separates model execution from graph overhead without excluding either from the headline number.

Hardware and software lock. The primary latency experiment uses one NVIDIA H100 80GB SXM, CUDA 12.x, PyTorch 2.x, BF16 weights, FlashAttention where supported, and identical tokenizer/model revisions for all DreamReasoner-8B variants. GPU clocks are left at cluster defaults but recorded. Each process receives a dedicated GPU; no competing job may share the device. Comparisons that require a different serving engine are repeated with compilation disabled to rule out engine-specific advantage.

Timing boundary. The timer starts after raw text is loaded but *before* tokenization, semantic-graph prediction, verifier execution, branch management, or diffusion scheduling, and stops after answer parsing. Thus the headline metric is truly end to end. We additionally decompose time into tokenization, backbone forward passes, structure head, scheduler, verifier, repair, and answer parsing. Synchronization calls are issued immediately before the start and stop boundaries. For each condition, 50 requests are used for warm-up and are discarded; the following 500 requests are timed individually.

Quality–latency frontier. Figure 3 reports the DAGReason accuracy–latency frontier at batch size one using the end-to-end timing boundary above. Table 1 is a separate endpoint-quality study.

Aggregation. Headline latency is the median end-to-end latency across timed requests. Speedup is computed per paired instance before aggregation.