SMARTER THAN RANDOM, OR SIMPLY BUSIER? WHAT CHOICE IS WORTH IN DIFFUSION-LLM DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models unmask several tokens per denoising step, and inference-time schedulers decide which positions to unmask. Every published scheduler reports a gain over a baseline that applies no rule. Such a comparison cannot tell whether the gain comes from choosing the right tokens or from decoding a different number of them per step. Thus, we introduce the matched-action control, which decodes the same number of tokens as the rule at each step at randomly chosen positions. Against this control, a reported gain splits into a substrate, the value of the confidence ordering every scheduler shares, and an overlay, the value of what each rule adds. On base and instruction-tuned models the substrate is worth $+0.011$ to $+0.016$ on open-ended generation and $+0.07$ to $+0.22$ on GSM8K and WikiSQL, and the overlay from choosing which tokens to decode is close to nothing. The reason is that confidence ordering already maximizes the expected number of correct tokens at a fixed number of unmasked positions under the model's own distributions. A rule can improve on it only through dependence among the tokens decoded together, and the improvement is bounded by how many of those tokens would change if they were unmasked one at a time. The bound is far from zero, but the delay rule recovers only 4% to 13% of it, because most of the changes it counts do not affect correctness and the rest are no more often right than wrong. What remains is the dependence itself, which is created by unmasking many positions at once on little context. Choosing how many tokens to unmask per step, and how far apart, is where the value lies. Unmasking few positions early and spacing the positions unmasked together, with no criterion and no extra cost, improves GSM8K accuracy by $+0.05$ to $+0.29$ over standard confidence decoding on LLaDA-8B and Dream-7B, base and instruction-tuned.

1 INTRODUCTION

Masked diffusion models for discrete data (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024) have recently scaled into competitive language models (Nie et al., 2025a; Ye et al., 2025). Unlike an autoregressive model, which produces one token per forward pass, a masked diffusion model holds a distribution over every masked position at once and can unmask several of them per pass. At each step it must decide which of the remaining positions to fill. A fast-growing literature answers with decoding strategies that decide which positions to unmask and how many, and each reports an improvement over decoding without it, in accuracy at a given number of steps or in steps at a given accuracy (Wu et al., 2026; Hong et al., 2026; Luxembourg et al., 2026; Abu Ayoub et al., 2026).

Consider what such an improvement shows. Two explanations fit equally well. The rule may be choosing the right tokens to decode, as claimed. Or decoding that many tokens per step may help no matter which ones they are, in which case the gain belongs to the schedule and would survive with the selection replaced by randomness. The published comparison cannot tell these apart, because the rule and its baseline differ both in how many tokens they decode per step and in which.

A hyperparameter search method is judged against random search at the same budget, because beating no search is not evidence of searching well (Bergstra & Bengio, 2012). The counterpart for

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.

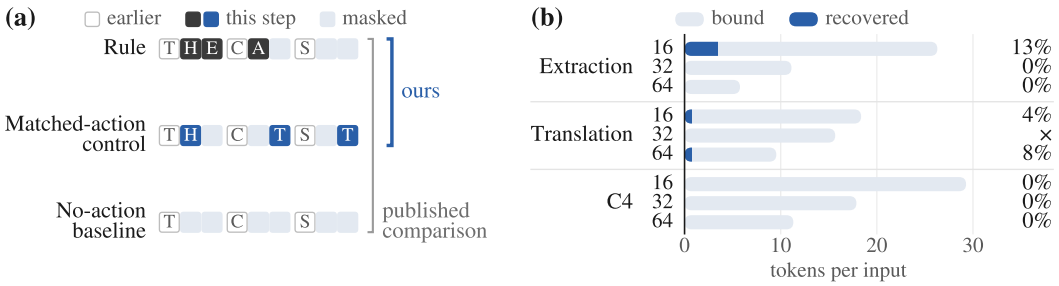


Figure 1: (a) One decoding step under three schedules. Published methods compare a rule against a baseline that applies no rule. We compare it against a matched-action control that decodes the same number of tokens at each step at randomly chosen positions, so the two runs differ only in which tokens. **(b)** The divergence bound in tokens per input (grey) and the part the delay rule recovers (blue), with the recovered share beside each row, LLaDA-8B at the step count shown. A share whose interval includes zero is drawn as 0%.

scheduling rules is a control that decodes the same number of tokens as the rule at each step, at randomly chosen positions, which we call a **matched-action control** (Figure 1a). To our knowledge, none of the published rules has been compared against one. Without it, a reported gain cannot tell a rule that is smarter than random (chooses the right tokens) from one that is simply busier (decodes a different number of them per step).

Under this control we evaluate two rules on two base models, five tasks and three step counts, and again on the instruction-tuned versions of both models decoded in blocks of 32 positions. Every rule adds its own criterion on top of the same confidence-ordered schedule, and we separate the two contributions as a **substrate** (what the shared confidence ordering is worth over random positions at the same number of tokens per step) and an **overlay** (what the rule's own criterion adds on top of that ordering). The substrate accounts for the gain, a consistent $+0.013$ on open-ended generation (C4) on both models and between $+0.10$ and $+0.22$ on WikiSQL extraction and GSM8K, and it holds on the instruction-tuned models, $+0.07$ to $+0.18$ on the same two tasks. The overlay carries almost none. On open-ended generation it adds nothing at any step count we tried, to a resolution of 0.0016, and on reasoning and extraction it is small and short-lived, up to $+0.076$. The threshold rule, whose criterion sets the number of positions a step unmasks, is the exception, and it improves WikiSQL by $+0.08$ to $+0.21$ on both base models.

This pattern has a simple explanation. At each step the model assigns a distribution to every masked position, and unmasking the positions with the highest confidence already maximizes the expected number of correct tokens under those distributions. A rule that unmasks differently can therefore improve on this only if the tokens decoded together are statistically dependent, so that decoding them one at a time would have changed some of them, and the number that would change is then an **upper bound** on what any rule can add at a fixed number of unmasked positions. We estimate this bound from one extra pass along the decoder's own trajectory (Figure 1b), and it is substantial even on open-ended generation. Yet most of the token changes it counts leave correctness untouched, and the rest are no more often for the better than for the worse, so the delay rule recovers only 4% to 13% of the bound where it improves at all. In short, dependence is necessary for a rule to gain, and it is far from sufficient.

The dependence above is created by the decode itself, when many positions are unmasked together on little context, so how much of it a step creates depends on how many positions the step fills and on how far apart they lie. The first choice gives **slow start**, which unmasks few positions in the early steps and more in the later ones at the same total number of forward passes, and improves the base models on WikiSQL and GSM8K by up to $+0.21$. It also moves most of the unmasking into late steps that each fill many positions, and there the second choice takes over. Spacing the positions a step unmasks, so that no two lie within two of each other, changes little when a step fills few positions but helps when it fills many. On the instruction-tuned models slow start alone lowers accuracy on LLaDA and leaves Dream unchanged, and slow start with spacing raises it. Meanwhile, the combination needs no criterion, no labels and no extra computation.

Our contributions are as follows.

- **A matched-action control** for scheduling rules, whose cost in forward passes is identical to the rule's on every input.
- **The substrate/overlay decomposition**, which attributes the value of parallel decoding strategies to the confidence ordering all rules share rather than to the criteria they add for choosing which tokens to decode.
- **An upper bound** on what any rule can add at a fixed number of unmasked positions, estimated from one extra pass over the plain decoder's own trajectory, with the finding that the delay rule recovers only 4% to 13% of it.
- **Slow start with spaced confidence ordering**, a schedule with no criterion and no extra cost that beats standard confidence decoding in every held-out setting we ran and matches dilated scheduling at block length 32, with a decomposition showing which of its two ingredients helps where.

On the translation task, where a model can score well by copying the source sentence from the prompt, a second control against such copying set aside one of our own statistically significant results.

2 Where a Scheduler's Gain Comes From

2.1 Setup

A masked diffusion language model receives a sequence in which a set M_0 of positions is masked. At each step $t = 1, \ldots, T$ it produces a distribution at every masked position. A schedule then unmasks a set $C_t \subseteq M_{t-1}$ of positions, meaning their tokens are fixed, and the remaining masked set becomes $M_t = M_{t-1} \setminus C_t$. Decoding ends when no masked positions remain. Decoding is greedy throughout, and cost is measured in forward passes.

Every rule we evaluate starts from plain decoding, which unmasks positions in confidence order, and applies an added criterion on top. The criteria are stated in §4. We separate a rule's behaviour into a gate, whether the criterion is applied at this step, and a selection, which tokens it applies to when it is. When the gate stays closed the step proceeds as plain confidence-ordered decoding, so the gate never halts decoding, it only withholds the criterion. The number of tokens the criterion applies to at step t, k_t, is the step's count, and $(k_1, \ldots, k_T)$ is its per-step profile.

2.2 Two ways to beat a baseline

A rule that beats a baseline decoding without it may have chosen the right tokens or may merely have decoded a different number of them per step. Separating the two takes two comparisons, one that holds the number of tokens per step fixed and asks whether they were well chosen, and one that holds the total cost fixed and asks whether the criterion was worth applying at all.

Definition 1 (Substrate and overlay)**.** *The **step-matched comparison** sets a rule against a control that decodes the same number of tokens as the rule at each step, at randomly chosen positions. The **total-matched comparison** sets it against the schedule without its criterion at the same total number of forward passes. The **substrate** is what the step-matched comparison measures for a confidence-ordered schedule, the value of unmasking in confidence order rather than at random. The **overlay** is a criterion's contribution beyond that ordering. For the delay rule, which chooses which positions to unmask, it is measured in the step-matched comparison, and for the threshold rule, which chooses how many, it is measured in the total-matched comparison, so its overlay is the value of an adaptive width rather than of a position choice.*

The two comparisons receive different answers in §5, and conflating them attributes a substrate gain to an overlay. The first comparison needs a control that decodes the same number of tokens as the rule at each step, at random positions. §3 constructs it and shows that it costs the same forward passes on every input.

2.3 WHAT IS LEFT TO CHOOSE

When several positions are unmasked in one step, each is fixed to its own most probable token, so how many of them come out right depends only on the model's per-position distributions. Under those distributions, the best set of a given size is the most confident one.

Proposition 1 (Confidence ordering is optimal)**.** *Write p_i for the model's distribution at masked position i, $\hat{t}_i$ for its most probable token and q_i for the true marginal. When a set C of k positions is unmasked together, the expected number of correct tokens is $\sum_{i \in C} q_i(\hat{t}_i)$, which does not depend on the joint distribution over C. If $q_i = p_i$, the sum is maximised over all sets of size k by the k positions with the largest $\max_t p_i(t)$.*

What the proposition leaves out is the joint distribution. It enters only when positions are unmasked one after another, because unmasking i first lets the model revise its distribution at j on the token it has just seen. That revision is the only source of improvement a criterion can have over confidence ordering at a fixed number of tokens decoded per step, and it is bounded in size by how often it changes the most probable token.

Proposition 2 (Divergence bound)**.** *Let Δ be the improvement in expected correct tokens that any criterion can obtain at a step over confidence ordering at the same number of tokens decoded per step. Then*

$$\Delta \;\le\; \sum_{j \in C} \Pr\big[\, j\text{'s most probable token changes given } t_{<j} \,\big],$$

where $t_{<j}$ are the tokens of C decoded before j when C is decoded one at a time in confidence order. The right-hand side is the number of tokens in C that would change under that sequential decode, which is what §6.1 measures. We call it the ***divergence bound****.*

Two consequences follow. With one token decoded per step the bound is zero, and no criterion can improve on confidence ordering. And the bound holds under the true joint distribution, so where the model's revised distributions are no more accurate than its marginals, little of it is attainable, and §6 measures how little, with the model's own marginals standing in for the data's. Proofs, a toy model in which the bound is tight, and the two caveats the bound carries, calibration and the task metric's dependence on token correctness, are in Appendix A.9.

3 HOLDING EVERYTHING ELSE EQUAL

The comparisons of §2 need a control that decodes the same number of tokens as the rule at each step, at random positions, a measurement of the bound that leaves the run it observes untouched, and two safeguards that keep both honest.

3.1 THE MATCHED-ACTION CONTROL

To control a rule, we run it first, recording its gate decisions and per-step counts $(k_1, \ldots, k_T)$ on each input. The control replays that record on the same input, applying the count at every open step to k_t tokens drawn uniformly at random from those still masked and doing nothing at closed steps, so the two runs change the same number of tokens at the same steps and differ only in which ones. It replays the recorded profile rather than running the criterion on shuffled confidence maps, because a criterion fed shuffled maps does less (the delay rule defers a median of 41.5 tokens on real maps and 12 on shuffled ones). It draws uniformly rather than from the band the rule favours, because knowing which tokens to defer is the rule's contribution and not an overhead to equalise. And it matches step by step rather than in total, which freezes the timing (Appendix A.4). Because both runs decode the same number of tokens at every step they spend the same forward count on every input, which matters at gains of $+0.013$, where a control that used even a few forward passes fewer would leave every result open to the objection that the rule simply spent more compute.

3.2 MEASURING THE BOUND

The bound is measured on a side pass that reads the decoding state and never writes to it. At each step, on a copy of the state, the positions about to be unmasked together are unmasked one at a

time in confidence order with a forward pass after each, and the number whose token differs from the parallel decode is the step's divergence. Its sum over steps is the input's bound. The main run is asserted byte-identical with and without the side pass, and the bound is asserted zero when one position is unmasked per step.

3.3 TWO SAFEGUARDS

On translation the prompt contains the source sentence, and reproducing it gains partial credit from a token-overlap metric, so a rule can appear to translate better while copying more. We score every run against the reference and against the source and call the difference its **copy margin**. A comparison is read only when its two margins fall within 0.02 of each other and carry the same sign, a line fixed before any comparison it decides on magnitude (Appendix A.5).

Every contrast is paired on the same inputs, with 95% percentile intervals from 2000 bootstrap resamples. Beside every effect we report the smallest effect its sample reliably detects, the smallest synthetic effect recovered at least 80% of the time when added to the per-input differences. An effect below it is reported as unresolved rather than as evidence of absence (Card et al., 2020). An effect above it whose interval lies above zero is reported as resolved.

4 EXPERIMENTAL SETUP

Models. LLaDA-8B and Dream-7B, both base models, decoded greedily at 16, 32 and 64 denoising steps for a fully masked 256-token block, that is, 16, 8 and 4 positions per step. The block is decoded whole, the setting LLaDA reports for base models (Nie et al., 2025a), and the step counts span the range of tokens per step at which published methods operate. We read the results again on LLaDA-8B-Instruct and Dream-7B-Instruct, decoded semi-autoregressively in blocks of 32 positions, the setting published methods use for instruction-tuned models, at four to sixteen steps per block.

Tasks. Five tasks span conditional to open-ended generation, arithmetic (GSM8K), English-to-German translation (OPUS-100), structured extraction (WikiSQL), code (HumanEval) and prose continuation (PG-19, WikiText, C4), scored in $[0, 1]$ by exact match, token overlap, field accuracy, unit tests and token overlap respectively. Conditional tasks place the prompt before a fully masked block. Prose scatters masks through a text window. A cell is read only when plain decoding scores above a capability floor fixed in advance, 0.02 for arithmetic and 0.10 elsewhere, and every task uses one prompt format throughout (Appendix A.3).

Rules. Two criteria on top of confidence ordering. The threshold rule (Wu et al., 2026) unmasks every position whose confidence clears $\tau \in \{0.9, 0.7\}$, with the most confident position always unmasked so that decoding advances. The delay rule defers the less confident member of each adjacent pair among the tokens plain decoding would decode, our construction of the principle behind dilated scheduling (Luxembourg et al., 2026) applied inside confidence order. The threshold rule sets its own step count and is measured in the total-matched comparison. The delay rule runs at a fixed step count and is measured in the step-matched comparison. All rules run on a uniform schedule that unmasks the same number of positions at every step. **Spaced confidence ordering** takes a step's positions in descending confidence and skips any within $d = 2$ of one already taken in the same step, filling from the skipped positions, again by confidence, only when too few remain. §6.4 also evaluates three schedules with no criterion, slow start, slow start with spaced confidence ordering, and dilated scheduling (Luxembourg et al., 2026).

5 THE ORDERING WAS DOING THE WORK

This section places the comparison that published methods make next to the one our control makes, on the same runs. Take confidence-threshold decoding (Wu et al., 2026) on open-ended generation with Dream. It reaches 0.1299 token overlap, and a control that decodes the same number of tokens per step at random positions reaches 0.1163, so the rule appears to improve accuracy by $+0.014$. But decoding tokens in order of confidence with no rule at all reaches 0.1306. The gain was there before the rule was added. Table 1 repeats this decomposition for every rule and model, and Table 3 in Appendix A.1 for every step count. Every overlay in this section is a rule measured against a

Table 1: The substrate (confidence order against random order) and each rule's overlay. Left, base models, whole block, 16 steps, $\tau = 0.9$. Right, instruction-tuned models in blocks of 32 at 32 steps, the delay rule at 64 steps (the fewest at which it can act within a block), C4 whole at 16 steps, and the threshold rule against uniform decoding at its own forward count on each input. ✓ gain, × loss (interval clear of zero and effect over its detection threshold), ∘ no detectable effect, — not read. Intervals, $\tau = 0.7$, translation and every step count are in Appendix A.1.

	Task	LLaDA	Dream
Substrate	C4	$+0.013$ ✓	$+0.014$ ✓
	Extraction	$+0.106$ ✓	$+0.220$ ✓
	Arithmetic	$+0.197$ ✓	$+0.170$ ✓
Delay	C4	-0.000 ∘	-0.001 ∘
	Extraction	$+0.061$ ✓	$+0.066$ ✓
	Arithmetic	—	$+0.024$ ∘
Threshold	C4	—	—
	Extraction	$+0.095$ ✓	$+0.085$ ✓
	Arithmetic	$+0.059$ ✓	$+0.004$ ∘

	Task	LLaDA-Inst.	Dream-Inst.
Substrate	C4	$+0.011$ ✓	$+0.016$ ✓
	Extraction	$+0.177$ ✓	$+0.081$ ✓
	Arithmetic	$+0.183$ ✓	$+0.066$ ✓
Delay	C4	-0.001 ∘	-0.002 ×
	Extraction	$+0.031$ ✓	$+0.013$ ∘
	Arithmetic	$+0.064$ ✓	$+0.064$ ✓
Threshold	C4	$+0.002$ ∘	-0.000 ∘
	Extraction	-0.033 ∘	$+0.018$ ∘
	Arithmetic	$+0.084$ ✓	$+0.198$ ✓

control that differs from it in one thing only, the delay rule against its matched-action control and the threshold rule against the same decode without its criterion at the same number of forward passes. Every substrate is confidence ordering measured against random ordering with the same number of tokens per step.

Confidence-based ordering accounts for the gain. Decoding tokens in order of confidence rather than at random improves accuracy by about $+0.013$ on open-ended generation on both models, and by $+0.10$ to $+0.22$ on WikiSQL extraction and GSM8K across both models and both thresholds, with every interval clear of zero. On the instruction-tuned models, decoded in blocks of 32, it is worth $+0.07$ to $+0.18$ on the same two tasks and $+0.011$ and $+0.016$ on C4 (Table 1, right). This is not a starting point that a better choice of tokens improves on. It is the entire benefit of choosing which tokens to decode.

Choosing which tokens to decode adds nothing detectable on open-ended generation, and little, briefly, on reasoning and extraction. On C4 and PG-19, on both models and at every step count, the delay rule never improves accuracy, and our largest sample, 451 inputs on C4, can detect an effect of 0.0016 and measures one nearly fifty times smaller. On OPUS translation and WikiSQL extraction the delay rule improves LLaDA by $+0.076$ and $+0.061$ at 16 steps, and by 32 steps neither improvement survives. On the instruction-tuned models it adds nothing on C4, where Dream even loses 0.002, and gains a little in blocks, $+0.064$ on GSM8K at 64 steps on both models.

Confidence-threshold decoding is different in kind. Its criterion changes how many tokens each step decodes, not which, and it improves WikiSQL by $+0.09$ and $+0.21$ on LLaDA and by $+0.08$ and $+0.13$ on Dream at its two thresholds, and GSM8K on LLaDA but not on Dream. On the instruction-tuned models in blocks it improves GSM8K by $+0.08$ and $+0.20$ and leaves WikiSQL unchanged.

The improvement fades because parallel decoding does less damage at lower parallelism, not because the rule stops acting. The delay rule defers about 70 tokens per input at 64 steps, against 88 at 16, yet added to the same decode on GSM8K it turns from $+0.033$ at 32 steps to -0.040 at 64. At 4 tokens per step few are decoded together, and deferring one only puts off a token the model would have decoded correctly.

6 What Is Left to Choose, Measured

Proposition 2 caps what any rule can add at a fixed number of tokens per step by the number of tokens that would change if the step's positions were unmasked one at a time. We measure this bound on every task, model and step count of Table 5, ask how much of it the delay rule recovers and why the rest is out of reach, and then turn to the choices that do improve accuracy. Full tables are in Appendix A.1.

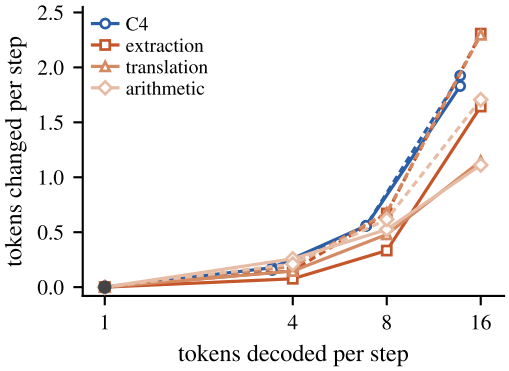


Figure 2: The divergence bound per step against tokens decoded per step, four tasks, two models. It is zero at one token per step and grows faster than linearly beyond.

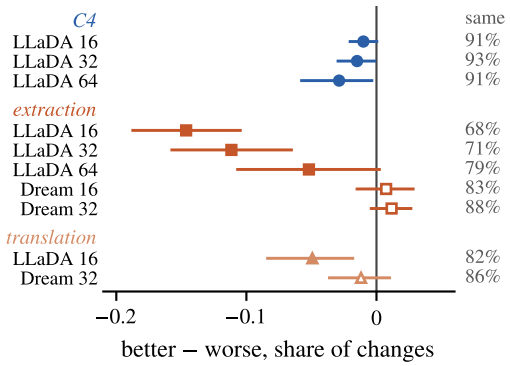


Figure 3: What the changes the bound counts are worth. Better minus worse under sequential decoding, as a share of the tokens that change, with 95% intervals, and at right the share that is as correct either way. No interval lies above zero.

6.1 The bound is real

The bound counts, for each step, how many of the tokens decoded together would have come out differently had they been decoded one at a time, which makes it a direct measure of what parallel decoding costs at that step, in tokens. Read this way, it behaves exactly as the proposition says, starting at zero when one token is unmasked per step, which we verify at 256 steps, and then growing faster than linearly with the number of tokens decoded together. On C4 with LLaDA, for instance, 0.18 of the tokens decoded together per step would have changed at about four tokens per step, 0.56 at seven and 1.8 at fourteen (Figure 2), and within every task the per-step cost falls monotonically as the step count rises. Nor is it small on open-ended generation. At 16 steps it adds up to 18 tokens per input on translation, 26 on extraction and 29 on C4, because every masked position of a prose window carries content, whereas most of a conditional task's block is padding that never changes. On the instruction-tuned models, by contrast, it is only 1 to 5 tokens per input on WikiSQL, blockwise or whole-block, while it stays at 29 to 31 on C4.

6.2 The rules recover little of it

To see how much of this bound a rule recovers, we convert the delay rule's gain into the bound's units, tokens per input that match the reference, and divide by the bound on the same inputs. Where the rule improves a task, that fraction is 4% to 13%, with 13% on extraction with LLaDA at 16 steps, 8% with Dream, and 4% and 5% on translation. This is not because the bound lies out of the rule's reach. Between 45% and 94% of it lies on adjacent positions, exactly where the rule acts, yet the rule recovers at most 9% of that part.

6.3 Why the rest is not attainable

The delay rule does target the right positions. Of the positions whose token would change under sequential unmasking, it defers 23% to 65% on the conditional tasks. But most of what it defers would not have changed, since only 2% to 17% of its deferrals are of that kind, and that share falls further as fewer tokens are decoded together. Moreover, even the changes that do occur are rarely improvements. Of them, 68% to 93% are as correct either way, and among the rest better minus worse is above zero in no cell and is -0.146 $[-0.188, -0.104]$ on extraction with LLaDA at 16 steps (Figure 3). A rule built on the same diagnosis confirms this. Deferring any token that would repeat an adjacent one cuts the repetition rate to a seventh of its control's and improves accuracy in no setting. On the instruction-tuned models the picture is simpler still, since in whole-block decoding the delay rule defers only 0% to 7% of the positions that would change and there is almost nothing for it to recover.

Table 2: Left, gains over uniform confidence decoding (U) on GSM8K at equal forward passes in blocks of 32 and 64, for slow start (SS), spacing at uniform steps (U-sp), slow start with spacing (SS-sp) and dilated scheduling (DUS). Right, dilated scheduling split into its schedule (its schedule in confidence order minus U), its spacing (DUS minus its schedule in confidence order) and DUS minus its schedule at random positions, base models over a whole block and instruction-tuned models inside blocks of 32. † the authors' code. Bold, interval clear of zero and effect over its detection threshold. —, not read (Appendix A.5).

	Model	B	SS	U-sp	SS-sp	DUS
Base	LLaDA	32	+0.021	−0.001	**+0.044**	**+0.045**
		64	+0.014	+0.011	**+0.050**	**+0.039**
	Dream	32	+0.015	+0.020	**+0.115**	+0.038
		64	—	—	—	—
Instruct	LLaDA	32	**−0.055**	−0.033	**+0.097**	**+0.111**
		64	**−0.149**	**+0.086**	**+0.141**	**+0.220**
	Dream	32	−0.001	+0.034	**+0.297**	**+0.309**
		64	−0.027	**+0.096**	**+0.206**	**+0.283**

	Task	Model	Schedule	Spacing	vs. random
Base	Extraction	LLaDA	**+0.306**	+0.008	**+0.099**
		Dream	—	—	—
	Arithmetic	LLaDA	—	—	—
		Dream	+0.012	+0.005	+0.001
Instruct	Arithmetic	LLaDA	**−0.190**	**+0.301**	**+0.069**
		Dream	**−0.133**	**+0.442**	**+0.115**
		LLaDA†	**−0.193**	**+0.428**	**+0.103**

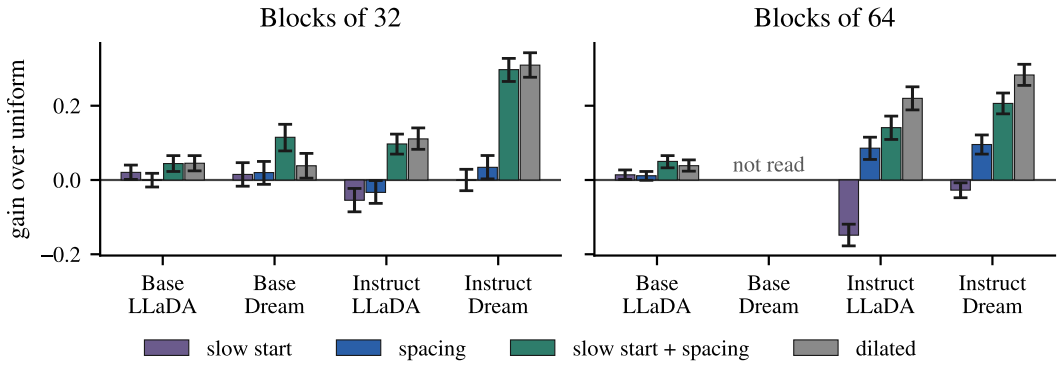


Figure 4: GSM8K gains over uniform confidence decoding at equal forward passes, with 95% intervals, the left half of Table 2. Base Dream at block length 64 is not read because its uniform decode does not clear the capability floor.

6.4 How many, and how far apart

Few tokens first. If the damage is done in the first steps, when many tokens are decoded together on almost no context, the natural remedy is to decode fewer of them at the start. Slow start is a coarse-to-fine unmasking schedule that unmasks 10% of the block over the first quarter of the steps and the rest on a linear ramp, at the same total forward passes as uniform decoding. On the base models, decoding the whole block, it improves extraction by +0.148 on LLaDA and +0.171 on Dream at 16 steps and +0.075 and +0.213 at 64, and GSM8K by +0.060 on Dream and by +0.063 and +0.051 on LLaDA at 32 and 64 steps, where the delay rule of §5 no longer helps (Table 6). The coarse-to-fine schedule of dilated scheduling does the same, +0.306 on extraction with LLaDA over the whole block. On the instruction-tuned models, however, the schedule alone is a loss. In blocks of 32, the dilated schedule with positions chosen by confidence lowers GSM8K by 0.190 and 0.133, and by 0.193 in its authors' own code (Table 2, right). The reason is visible in the schedule itself. A coarse-to-fine schedule moves most of the decoding into the last steps of each block, half of the block at once under the dilated schedule.

Spaced apart. Spacing the positions a step unmasks (§4) changes that. Under the dilated schedule inside blocks of 32 it raises GSM8K by +0.238 and +0.298 on the instruction-tuned models and by +0.049 and +0.107 on the base models, whereas on WikiSQL it does nothing on either, −0.027 to +0.020. The spacing also has to be wide enough, since a spacing of one instead of two recovers less, +0.174 and +0.165 on GSM8K.

Together, and held out. The two ingredients therefore interact. In blocks of 32, at uniform steps spacing moves GSM8K by −0.033 and +0.034 on the instruction-tuned models, and slow start alone by −0.055 and −0.001, while slow start with spaced confidence ordering raises it by +0.097 and +0.297 (Figure 4). At block length 64, where every uniform step fills ten or eleven positions, spacing

alone already helps, $+0.086$ and $+0.096$, and the combination gives $+0.141$ and $+0.206$. We then fixed this schedule and tested it where its selection could not see, on 1000 GSM8K training items at block length 32 and on the GSM8K test items at block length 64. It beats standard confidence decoding in all seven readable settings, by $+0.05$ to $+0.29$. Against dilated scheduling it holds even at block length 32, from -0.018 to $+0.077$ on the four settings, and at block length 64 it matches it on the base model we can read but trails it by 0.079 and 0.077 on the instruction-tuned ones. The block-64 setting reuses the test items that selected the schedule at block length 32, so it is held out in block length only.

Dilated scheduling. The same decomposition explains where dilated scheduling's own gain comes from. On the base models its schedule carries it, $+0.306$ on extraction, and its spacing leaves it unchanged, $+0.008$. On the instruction-tuned models inside blocks, by contrast, its spacing carries it, $+0.301$ and $+0.442$ over confidence order under the same schedule, and $+0.428$ in its authors' code, and the advantage of its fixed spacing over ours grows with the block. Why confidence order fails under this schedule on instruction-tuned models we have not located, since the positions it leaves for a block's last step are adjacent as often on base-model extraction, where it does not fail (Appendix A.2 gives this and the same decomposition for whole-block and block-wise decoding).

7 Related Work

Decoding strategies for masked diffusion LLMs began with planners that rank positions by the model's own confidence, a construction masked image generation arrived at first (Chang et al., 2022; Nie et al., 2025a), and the field has since looked for further gains in four places. Some spend extra forward passes to inspect the joint distribution before decoding (Lee et al., 2025), which the bound of §2 caps at a fixed number of tokens per step by a quantity that is mostly unattainable on current models. Some hand the selection to a planner, external or the denoiser itself (Peng et al., 2025), or train the model to decode more tokens per step (Chen et al., 2026), and these choose positions under the same cap. Some allow a decoded token to be remasked (Wang et al., 2025; Hong et al., 2026), repairing after the fact the damage the first steps do. And some set how many tokens each step unmasks (Ben-Hamu et al., 2025; Park et al., 2025; Luxembourg et al., 2026; Cai & Li, 2026), where our gains turn up. Dilated scheduling combines two ingredients that help in different settings, its schedule on base models and its spacing on instruction-tuned ones.

On the theory side, Cai & Li (2026) show that confidence-based unmasking is efficient in iterations, which our Proposition 1 complements with optimality in expected accuracy. Dilated scheduling motivates its spacing by the same gap between marginal and joint entropy that our bound turns around to cap any choice of positions. Kim et al. (2025) show that on strongly structured tasks such as Sudoku the unmasking order is decisive, and our conclusions are stated for tasks where it is not. Random controls at matched cost are the standard remedy when a gain could come from resources rather than choices (Bergstra & Bengio, 2012; Li & Talwalkar, 2019).

8 Conclusion

What choice is worth in diffusion-LLM decoding depends on how many tokens a step decodes. When few, the ordering is all that matters, and every rule we tested that chooses which tokens to decode on top of it adds little or nothing. When many, what matters is how many to decode and how far apart, since confidence alone then leaves adjacent, dependent tokens to be decoded together, and a schedule that unmasks few tokens early and spaces the rest apart recovers most of the loss at no extra cost. These conclusions come with limits. On instruction-tuned models decoded as one whole block, confidence ordering decodes end-of-sequence tokens too early, so our readings for those models are stated for blocks of 32 (Appendix A.2), and our step counts are fixed in advance rather than taken from the published methods' own operating points. Spaced confidence ordering has been held out only on GSM8K, where on WikiSQL its spacing added nothing, and at block length 64 the fixed spacing of dilated scheduling still beats ours on the instruction-tuned models by 0.08. Why confidence order fails under coarse-to-fine schedules on those models but not on base-model extraction is not settled by the adjacency or the token types we measured, and that, together with longer blocks and tasks beyond GSM8K, is where we would look next.

AI USE STATEMENT

We used a generative AI assistant (Anthropic's Claude, accessed through a coding agent) during this project. Its use is disclosed below.

Tasks with required disclosure, used. We used it to refine hypotheses, in particular to turn the two-layer decomposition of §2 into the divergence bound of Proposition 2 and to specify the slow start and spaced confidence ordering schedules of §6.4 before they were run. We used it to interpret results, in reading the budget measurements, the fraction of the bound recovered and the yield measurements of §5 and §6 and the instruction-tuned replication.

Tasks with required disclosure, not used. We did not use generative AI to generate synthetic datasets, to clean or reformat datasets, or to assist with translation. Formulating questions for surveys or interviews and transcribing recordings of research material are not applicable to this work.

Tasks with recommended disclosure. We also used generative AI to draft and edit parts of this paper, to create its figures and tables from the result files, to identify and summarise related literature, to format references, and to search for information.

Review procedures. We have reviewed all AI-assisted work. All AI-written code was read by the authors, every table and figure is generated by a script from the per-input result files, and every number in the text is checked by a guard against those files. The proofs of Propositions 1 and 2 were re-derived independently by the authors. Every bibliographic entry in this paper was verified against a public authoritative record, namely the publisher's page, the official proceedings, or arXiv, checking the title, the complete author list, the venue and the year. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference-time decoding of existing, publicly released masked diffusion language models and introduces no new data collection or training. All datasets used (GSM8K, OPUS-100, WikiSQL, HumanEval, MBPP, PG-19, WikiText, C4) are established public benchmarks used under their respective licences. We are not aware of ethical concerns specific to this work beyond those already inherent to the underlying models and benchmarks.

REPRODUCIBILITY STATEMENT

Every method in this paper is training-free and adds no parameters, so reproduction requires only the publicly released checkpoints of LLaDA-8B, Dream-7B and their instruction-tuned versions. §3 gives the construction of the matched-action control and the measurement of the bound, §4 the tasks, prompt formats, capability floors and rules, and Appendix A the datasets and their cleaning, the statistics and floors, and the full tables behind every figure; the preregistrations, with their frozen predictions and every deviation from them, are released with the code. All reported numbers are produced by scripts from per-input result files, each carrying the commit and seed that produced it. We will release the code, the result files and the evaluation configurations.

REFERENCES

Giries Abu Ayoub, Mario Barbara, Lluís Pastor-Pérez, Tanja Bien, Aneesh Barthakur, Alaa Maalouf, and Loay Mualem. Supportive token revealing for fast diffusion language model decoding, 2026. arXiv:2606.04236.

J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021. arXiv:2107.03006.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *NeurIPS*, volume 38, pp. 55981–56007, 2025. arXiv:2505.24857.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012. 13:281–305.

Changxiao Cai and Gen Li. Confidence-based decoding is provably efficient for diffusion language models, 2026. arXiv:2603.22248.

D. Card, P. Henderson, U. Khandelwal, R. Jia, K. Mahowald, and D. Jurafsky. With little power comes great responsibility. In *EMNLP*, 2020. arXiv:2010.06595.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *CVPR*, 2022. arXiv:2202.04200.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *ICLR*, 2026. arXiv:2509.26488.

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective DLLMs. In *ICLR*, 2026. arXiv:2507.18578.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *ICML*, 2025. arXiv:2502.06768.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models, 2025. arXiv:2511.05563.

L. Li and A. Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019. arXiv:1902.07638.

A. Lou, C. Meng, and S. Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, 2024. arXiv:2310.16834.

O. Luxembourg, H. Permuter, and E. Nachmani. Plan for speed: dilated scheduling for masked diffusion language models. In *ICML*, 2026. arXiv:2506.19037.

S. Nie, F. Zhu, Z. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and C. Li. Large language diffusion models. In *NeurIPS*, 2025a. arXiv:2502.09992.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *ICLR*, 2025b. arXiv:2410.18514.

Yong-Hyun Park, Chieh-Hsin Lai, Satoshi Hayakawa, Yuhta Takida, and Yuki Mitsufuji. Jump your steps: Optimizing sampling schedule of discrete diffusion models. In *ICLR*, 2025. arXiv:2410.07761.

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, 2025. arXiv:2502.03540.

S. S. Sahoo, M. Arriola, Y. Schiff, A. Gokaslan, E. Marroquin, J. T. Chiu, A. Rush, and V. Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024. arXiv:2406.07524.

J. Shi, K. Han, Z. Wang, A. Doucet, and M. K. Titsias. Simplified and generalized masked diffusion for discrete data. In *NeurIPS*, 2024. arXiv:2406.04329.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *NeurIPS*, 2025. arXiv:2503.00307.

C. Wu, H. Zhang, S. Xue, Z. Liu, S. Diao, L. Zhu, P. Luo, S. Han, and E. Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *ICLR*, 2026. arXiv:2505.22618.

J. Ye, Z. Xie, L. Zheng, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong. Dream 7b: Diffusion large language models, 2025. arXiv:2508.15487.

A Appendix

A.1 Full result grids

Tables 4 and 5 give every reading behind Tables 1 and 3, with its interval and the smallest effect its sample can detect. Figure 5 draws the substrate and the overlays on open-ended generation and translation for both models.

Table 3: The delay rule against its matched-action control on LLaDA-8B, the grid behind §5. ✓ gain, ∘ no detectable effect, — not read. Table 5 gives the intervals, the reasons behind each —, and the same grid on Dream-7B.

Task	Steps	Overlay	Detect.	
Arithmetic	16	—	—	—
	32	$+0.004$	0.025	∘
	64	-0.002	0.030	∘
Translation	16	$+0.076$	0.029	✓
	32	—	—	—
	64	$+0.017$	0.023	∘
Extraction	16	$+0.061$	0.043	✓
	32	$+0.019$	0.041	∘
	64	$+0.000$	0.033	∘
Prose (C4)	16	-0.000	0.002	∘
	32	-0.000	0.002	∘
	64	$+0.001$	0.001	∘

Table 4: The substrate and each rule's overlay at both thresholds and every step count, with 95% intervals. Bold as in Table 1. ‡ marks a reading whose rule acts on too few tokens to clear the activity floor of Appendix A.5, printed and not cited. — not read, because the copying control refused the comparison. The delay rule's readings on C4 are those of Table 5.

			LLaDA-8B		Dream-7B	
Layer / rule	Task	Setting	Effect	95% CI	Effect	95% CI
Substrate	C4	$\tau = 0.9$	$\mathbf{+0.0126}$	$[+0.0103, +0.0149]$	$\mathbf{+0.0136}$	$[+0.0108, +0.0162]$
		$\tau = 0.7$	$\mathbf{+0.0120}$	$[+0.0095, +0.0145]$	$\mathbf{+0.0130}$	$[+0.0106, +0.0155]$
Substrate	Extraction	$\tau = 0.9$	$\mathbf{+0.1060}$	$[+0.0796, +0.1326]$	$\mathbf{+0.2197}$	$[+0.1947, +0.2446]$
		$\tau = 0.7$	$\mathbf{+0.1014}$	$[+0.0751, +0.1274]$	$\mathbf{+0.1765}$	$[+0.1452, +0.2073]$
Substrate	Arithmetic	$\tau = 0.9$	$\mathbf{+0.1966}$	$[+0.1734, +0.2212]$	$\mathbf{+0.1701}$	$[+0.1457, +0.1941]$
		$\tau = 0.7$	$\mathbf{+0.1859}$	$[+0.1616, +0.2100]$	$\mathbf{+0.1496}$	$[+0.1274, +0.1736]$
Threshold	C4	$\tau = 0.9$	$+0.0009^{\ddagger}$	$[-0.0005, +0.0023]$	$-0.0007^{\ddagger}$	$[-0.0022, +0.0010]$
		$\tau = 0.7$	$+0.0009^{\ddagger}$	$[-0.0009, +0.0030]$	$-0.0000^{\ddagger}$	$[-0.0017, +0.0017]$
Threshold	Extraction	$\tau = 0.9$	$\mathbf{+0.0947}$	$[+0.0628, +0.1291]$	$\mathbf{+0.0850}$	$[+0.0585, +0.1142]$
		$\tau = 0.7$	$\mathbf{+0.2143}$	$[+0.1766, +0.2539]$	$\mathbf{+0.1322}$	$[+0.0937, +0.1710]$
Threshold	Arithmetic	$\tau = 0.9$	$\mathbf{+0.0591}$	$[+0.0295, +0.0886]$	$+0.0041$	$[-0.0263, +0.0345]$
		$\tau = 0.7$	$+0.0345$	$[+0.0033, +0.0648]$	-0.0008	$[-0.0304, +0.0263]$
Threshold	Translation	$\tau = 0.9$	—	—	$+0.0139$	$[-0.0074, +0.0348]$
		$\tau = 0.7$	—	—	$\mathbf{+0.0345}$	$[+0.0127, +0.0559]$
Attention	PG-19	16 steps	$+0.0000^{\ddagger}$	$[-0.0004, +0.0005]$		
Attention	C4	32 steps	$+0.0005^{\ddagger}$	$[-0.0003, +0.0014]$		
		64 steps	$+0.0022^{\ddagger}$	$[+0.0009, +0.0036]$		

Table 5: The delay rule against its matched-action control on both models, at every task and step count. ✓ and ○ as in Table 3. — not read, because the copying control refused the comparison ([c]) or plain decoding does not clear the capability floor ([f]), see Appendix A.5. Blank cells were not collected.

		LLaDA-8B				Dream-7B			
Task	Steps	Overlay	95% CI	Detect.		Overlay	95% CI	Detect.	
Arithmetic	16	+0.0161	[+0.0063, +0.0268]	0.0145	—[f]	+0.0239	[+0.0017, +0.0461]	0.0300	○
	32	+0.0041	[−0.0139, +0.0213]	0.0254	○	+0.0072	[−0.0217, +0.0344]	0.0417	○
	64	−0.0022	[−0.0227, +0.0186]	0.0298	○	+0.0122	[−0.0161, +0.0422]	0.0414	○
Translation	16	+0.0757	[+0.0538, +0.0955]	0.0290	✓	—	—	—	—[c]
	32	—	—	—	—[c]	+0.0414	[+0.0202, +0.0633]	0.0301	✓
	64	+0.0171	[+0.0012, +0.0330]	0.0234	○	—	—	—	—[c]
Extraction	16	+0.0608	[+0.0313, +0.0923]	0.0429	✓	+0.0661	[+0.0324, +0.0991]	0.0494	✓
	32	+0.0194	[−0.0083, +0.0477]	0.0408	○	+0.0179	[−0.0130, +0.0478]	0.0425	○
	64	+0.0002	[−0.0236, +0.0229]	0.0334	○	−0.0069	[−0.0364, +0.0234]	0.0443	○
Code	16	+0.0122	[−0.0061, +0.0346]	0.0224	—[f]				
	32	+0.0061	[−0.0102, +0.0244]	0.0263	—[f]				
	64	+0.0102	[−0.0061, +0.0305]	0.0262	—[f]				
Prose (C4)	16	−0.0000	[−0.0011, +0.0011]	0.0016	○	−0.0010	[−0.0022, +0.0004]	0.0018	○
	32	−0.0001	[−0.0012, +0.0010]	0.0016	○	−0.0008	[−0.0021, +0.0005]	0.0018	○
	64	+0.0006	[−0.0003, +0.0016]	0.0014	○	−0.0015	[−0.0025, −0.0005]	0.0015	○
PG-19	16	−0.0019	[−0.0036, −0.0003]	0.0025	○				

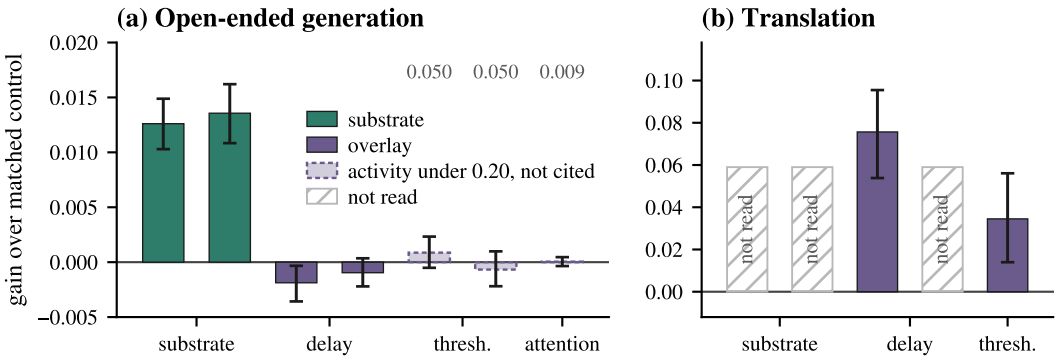


Figure 5: The layer split on both models on open-ended generation (C4, with PG-19 for LLaDA's delay and attention bars) and on translation. Within each rule the left bar is LLaDA and the right Dream; the attention rule was run on LLaDA only, and translation's threshold reading is Dream's. Hatched columns mark comparisons the copying control set aside, drawn at a fixed height. Faded bars with a dashed edge carry an activity below the floor of Appendix A.5, printed above them, and their nulls carry no conclusion.

A.2 Instruction-tuned models, spacing and dilated scheduling

This section holds the readings behind §5, §6.4 and §8 that concern the instruction-tuned models, the spacing of unmasked positions, and dilated scheduling, together with two decompositions the body only cites.

Which ingredient of the delay rule recovers the bound. The delay rule has two ingredients, deferring one token of each adjacent pair and choosing the less confident one. Which of the two accounts for the part of the bound it recovers varies from task to task and model to model, and for the threshold rule the corresponding split moves entirely from the schedule at $\tau = 0.9$ to per-input adaptivity at $\tau = 0.7$. No single property of the choice explains the gain across settings.

Slow start and dilated scheduling on the base models. Table 6 gives slow start's gain over uniform decoding on the base models at three step counts, and dilated scheduling at its own step counts over a whole block and over blocks. In its block rows the schedule and the random arms

chose positions over the whole mask while dilated scheduling walked its blocks, so the spacing and random columns there mix spacing with the choice of block. Table 7 repeats the reading with every arm choosing inside the block, and Figures 6 and 7 draw these gains with their intervals, on extraction and on GSM8K. On open-ended generation the linear ramp makes no difference, because the visible tokens scattered through the window already anchor the first steps, and on translation slow start cannot be read, because unmasking the most confident tokens first copies the source in the early steps and the copying control refuses every comparison.

Table 6: Base models. Left, slow start over the uniform schedule at three step counts, whole block. Right, dilated scheduling at its own step counts, split into its schedule and its spacing. In the block rows the schedule and random arms chose over the whole mask while dilated scheduling walked its blocks, and Table 7 repeats them with every arm inside the block. Bold, interval clear of zero and effect over its detection threshold. — not read (Appendix A.5). The halves are at different step counts and are not compared.

Task	Model	16	32	64
Extraction	LLaDA	**+0.148**	—	**+0.075**
	Dream	**+0.171**	—	**+0.213**
Arithmetic	LLaDA	—	**+0.063**	**+0.051**
	Dream	**+0.060**	—	+0.063
C4	LLaDA	+0.001	—	−0.000
	Dream	+0.002	—	+0.001
PG-19	LLaDA	−0.002	—	—
Translation	not read, both models			

Task	Model	Layout	Schedule	Spacing	vs. random
Extraction	LLaDA	whole	**+0.306**	+0.008	**+0.099**
		blocks	**+0.152**	**−0.052**	−0.000
	Dream	whole	—	—	—
		blocks	**+0.222**	**−0.155**	−0.040
Arithmetic	LLaDA	whole	—	—	—
		blocks	−0.022	**+0.048**	**+0.070**
	Dream	whole	+0.012	+0.005	+0.001
		blocks	−0.022	+0.010	**+0.089**
C4	LLaDA	whole	**+0.007**	**−0.024**	**−0.004**
		blocks	+0.001	**−0.016**	−0.001
	Dream	whole	**+0.007**	**−0.026**	**−0.007**
		blocks	+0.001	**−0.018**	−0.002
Translation	not read, both models and layouts				

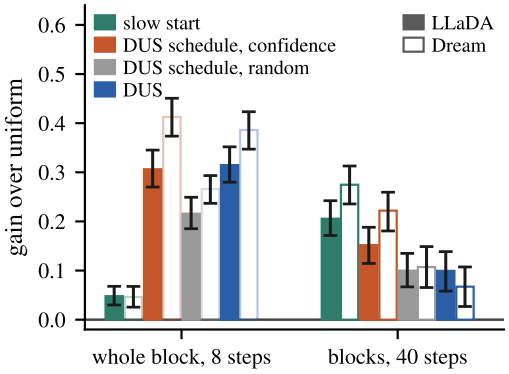


Figure 6: Gains over uniform confidence decoding on base-model extraction at equal forward passes, with 95% intervals. Pale bars are not read (Appendix A.5).

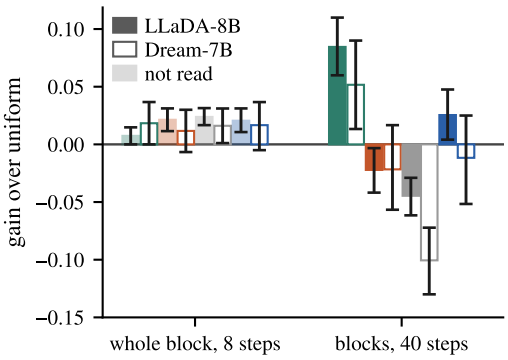


Figure 7: Gains over uniform confidence decoding on GSM8K at equal forward passes, with 95% intervals. Pale bars are not read (Appendix A.5).

Confidence against random under the same schedule. Table 7 holds dilated scheduling's schedule fixed and changes only how positions are chosen. R is random order minus confidence order, positive when confidence loses. On the base models confidence order beats random wherever the difference is resolved, in whole blocks and in blocks of 32 alike. On the instruction-tuned models it loses to random in both, and dilated spacing beats confidence order in every cell and random in six of seven. In the block rows every arm chooses inside the current block. The last group of rows is the setting of Table 6, in which confidence and random chose over the whole mask while dilated scheduling walked its blocks, so their spacing and random contrasts mix spacing with the choice of block and are kept only for comparison with Table 6.

Whole-block decoding of the instruction-tuned models. Decoded as one 256-token block, the instruction-tuned models decode end-of-sequence and padding tokens in the first quarter of the steps far more often under confidence order than under random order or dilated positions, in 13 of the 15 cells we diagnosed, and every cell in which the substrate or the threshold overlay turns into a loss

is one of them. Neither of the two cells without this signature, C4 with Dream and GSM8K with LLaDA at 16 steps, turns. This is why the body reads the instruction-tuned models in blocks of 32.

Spacing under every schedule. Table 8 gives every cell of the spacing collection, with a spacing of one position as well as two, and Table 9 the held-out confirmation of slow start with spacing, with its absolute scores.

Dilated scheduling's confidence threshold. The authors' released configuration adds a confidence threshold of 0.3 that returns a step's low-confidence tokens to the mask, never to be decoded again. With it dilated scheduling reads 0.270 on GSM8K; at a threshold of zero it reproduces the reported 0.657, and that is the configuration we run.

What the last step decodes. Under the dilated schedule a block's last step decodes half the block at once. In confidence order the positions left for that step are adjacent to one another in 0.88 to 0.96 of cases on every model and task, and under dilated positions in none, so adjacency at the last step is a property of confidence ordering rather than of the model. Format tokens among the tokens decoded early and adjacency at the last step each explain one of the four cells we measured, and neither explains why confidence order fails on the instruction-tuned models alone.

Table 7: Dilated scheduling's schedule with the choice of positions changed. R, random minus confidence; Spacing, dilated positions minus confidence; vs. random, dilated positions minus random. † the authors' code. Bold, resolved. —, not read (Appendix A.5).

Model	Decoding	Task		R	Spacing	vs. random
Base	whole	Arith.	LLaDA	—	—	—
Base	whole	Arith.	Dream	$+0.004$	$+0.005$	$+0.001$
Base	whole	Extr.	LLaDA	$\mathbf{-0.090}$	$+0.008$	$\mathbf{+0.099}$
Base	whole	Extr.	Dream	—	—	—
Base	in blocks	Extr.	LLaDA	$\mathbf{-0.092}$	$\mathbf{-0.055}$	$+0.037$
Base	in blocks	Extr.	Dream	$\mathbf{-0.161}$	$\mathbf{-0.145}$	$+0.016$
Base	blocks, global	Arith.	LLaDA	$\mathbf{-0.022}$	$\mathbf{+0.048}$	$\mathbf{+0.070}$
Base	blocks, global	Arith.	Dream	$\mathbf{-0.079}$	$+0.010$	$\mathbf{+0.089}$
Base	blocks, global	Extr.	LLaDA	$\mathbf{-0.052}$	$\mathbf{-0.052}$	-0.000
Base	blocks, global	Extr.	Dream	$\mathbf{-0.115}$	$\mathbf{-0.155}$	-0.040
Instruct	whole	Arith.	LLaDA	$\mathbf{+0.112}$	$\mathbf{+0.135}$	$+0.023$
Instruct	whole	Arith.	Dream	$\mathbf{+0.238}$	$\mathbf{+0.270}$	$\mathbf{+0.032}$
Instruct	whole	Extr.	LLaDA	$\mathbf{+0.175}$	$\mathbf{+0.300}$	$\mathbf{+0.125}$
Instruct	whole	Extr.	Dream	$\mathbf{+0.104}$	$\mathbf{+0.195}$	$\mathbf{+0.090}$
Instruct	in blocks	Arith.	LLaDA	$\mathbf{+0.232}$	$\mathbf{+0.301}$	$\mathbf{+0.069}$
Instruct	in blocks	Arith.	Dream	$\mathbf{+0.327}$	$\mathbf{+0.442}$	$\mathbf{+0.116}$
Instruct	in blocks	Arith.	LLaDA†	$\mathbf{+0.325}$	$\mathbf{+0.428}$	$\mathbf{+0.103}$
Instruct	blocks, global	Arith.	LLaDA	-0.011	$\mathbf{+0.218}$	$\mathbf{+0.230}$
Instruct	blocks, global	Arith.	Dream	$+0.021$	$\mathbf{+0.279}$	$\mathbf{+0.258}$
Instruct	blocks, global	Extr.	LLaDA	-0.001	$+0.027$	$+0.028$
Instruct	blocks, global	Extr.	Dream	-0.027	-0.014	$+0.013$

Table 8: Spacing inside blocks of 32, under the uniform schedule (U), slow start (SS) and the dilated schedule (P), against the same schedule in confidence order unless stated. Instruct: P-sp−P, P-sp−DUS, U-sp−U, SS-sp−U, then the first, third and fourth at a spacing of one. Base: P-sp−P, U-sp−U, SS-sp−SS. Bold, resolved.

	Task	Model	Contrasts, in the order of the caption						
Instruct	Arithmetic	LLaDA	$\mathbf{+0.238}$	$\mathbf{-0.063}$	-0.033	$\mathbf{+0.097}$	$\mathbf{+0.174}$	$+0.028$	$\mathbf{+0.064}$
Instruct	Arithmetic	Dream	$\mathbf{+0.298}$	$\mathbf{-0.144}$	$+0.034$	$\mathbf{+0.297}$	$\mathbf{+0.165}$	$\mathbf{+0.077}$	$\mathbf{+0.183}$
Instruct	Extraction	LLaDA	-0.005	-0.015	-0.016	$+0.023$	—	—	—
Instruct	Extraction	Dream	-0.021	$+0.025$	$\mathbf{-0.065}$	$\mathbf{+0.047}$	—	—	—
Base	Arithmetic	LLaDA	$\mathbf{+0.049}$	-0.001	$+0.024$				
Base	Arithmetic	Dream	$\mathbf{+0.107}$	$+0.020$	$\mathbf{+0.100}$				
Base	Extraction	LLaDA	$+0.020$	$+0.007$	-0.012				
Base	Extraction	Dream	-0.027	$+0.047$	$\mathbf{-0.095}$				

Table 9: Slow start with spacing held out: 1000 GSM8K training items at block length 32 and the test items at 64. Accuracy of uniform confidence decoding (U), slow start with spacing (SS-sp) and dilated scheduling (DUS), then SS-sp−U and SS-sp−DUS. Bold, resolved. —, not read (Appendix A.5).

Setting	Model	U	SS-sp	DUS	SS-sp−U	SS-sp−DUS
Train, B=32	Instruct LLaDA	0.635	0.717	0.724	**+0.082**	−0.007
Train, B=32	Instruct Dream	0.286	0.574	0.592	**+0.288**	−0.018
Train, B=32	Base LLaDA	0.057	0.124	0.136	**+0.067**	−0.012
Train, B=32	Base Dream	0.094	0.219	0.142	**+0.125**	**+0.077**
Test, B=64	Instruct LLaDA	0.394	0.535	0.614	**+0.141**	**−0.079**
Test, B=64	Instruct Dream	0.105	0.311	0.387	**+0.206**	**−0.077**
Test, B=64	Base LLaDA	0.021	0.070	0.059	**+0.050**	+0.011
Test, B=64	Base Dream	0.018	—	—	—	—

A.3 DATASETS

Arithmetic and code. GSM8K test items under one fixed prompt, scored by exact match on the final answer, and HumanEval and MBPP under their standard prompts, scored by unit tests on the delimited answer.

Translation. OPUS-100 German–English, test split, 2000 pairs. Two cleaning rules are applied before any sampling. Pairs whose two sides are identical are removed, 106 pairs, and pairs whose source is shorter than four words are removed, 322 pairs, their union being 383, which leaves 1617. Both rules exist to stop the task rewarding copying, and a pure copier still scores 0.071 at the median against the reference on this pool, the baseline any translation number must be read against. Translation is scored against one reference, so an adequate rendering unlike it is charged as an error.

Structured extraction. WikiSQL, with fields resolved through the table file. The task requires a demonstrated prompt, because under a bare prompt every arm scored identically. The prompt carries one demonstration, the same in every collection compared here. It is the harder of the two demonstrations we tried, and every extraction number in this paper is measured under it.

Open-ended text. PG-19 is chunked into non-overlapping 256-token windows, 3874 in all. C4 uses one validation shard of 45576 documents, each truncated to 40000 characters before tokenisation, giving 61381 windows at length 256. WikiText supplies the third corpus. A window masks 86% of its positions at random.

Mask layout on Dream. Dream predicts the token at position p from the logit row at $p-1$, so position 0 has no source row and can never be decoded. Masks for Dream are therefore drawn over positions $[1, L)$, about 0.86 fewer masked positions per window, so the two models' draws differ and no magnitude is compared across them.

A.4 HOW THE CONTROL WAS BUILT

Replay, not re-run. An apparently natural control runs the rule itself on shuffled inputs, so that it chooses at random. But how much a rule does when it fires depends on the statistics of its input. The delay rule triggers on adjacent pairs of high-confidence positions, and how many it finds depends on the confidence map it is given. On the same corpus at the same step count it defers a median of 41.5 tokens on real confidence maps and 12 on shuffled ones, with the gate opening about as often either way. The resulting run differs from the rule in amount, in timing and in position all at once, so the comparison recovers every confound the control was meant to remove. The control therefore replays the recorded per-step profile, matching the rule's behaviour rather than its mechanism.

Uniform, not calibrated. Making the random draw fairer, say by drawing only from the low-confidence band the rule favours, is tempting, but knowing which tokens to defer is not an overhead to equalise between runs. It is the rule's contribution. A control that is handed this knowledge measures only the residual value of choosing within the band, and the more carefully the draw is calibrated, the more of the signal it absorbs, until the measured effect can be tuned to zero by calibration alone. Uniform sampling is the one draw that encodes no knowledge of which tokens to defer, which is the null hypothesis the control exists to be.

Step by step, not in total. A rule chooses along two axes, which tokens to unmask and when. The delay rule, for instance, defers at its highest rate on the first step and at a nearly level rate thereafter, and its timing differs across tasks. A control matched only in total could spend the same forward passes on a different timing, and the comparison would then mix the value of choosing which tokens with the value of choosing when, in unknown shares. Replaying the profile step by step freezes the timing outright, leaving position choice as the only difference between the two runs. The total-matched comparison of §2 keeps timing in on purpose, since it prices the added criterion as a whole.

We assert on every input that the rule and its control spend the same forward count, together with agreement of their remaining-mask trajectories at every step, since equal totals alone would pass two runs that diverged and re-converged. Random draws are seeded per input, level and draw, and every random arm is the mean of three independent draws.

A.5 Statistics, floors and the copying line

Paired bootstrap. Every contrast in this paper is paired. The rule and its control run on the same inputs, and the reported effect is the mean of per-input differences. Intervals are 95% percentile intervals from 2000 bootstrap resamples of the inputs. Pairing removes the variance that inputs share, which matters at our effect sizes, where input-to-input score variation exceeds the effects by an order of magnitude.

Detection thresholds. A null result is only as strong as the effect the sample could have seen, so next to every effect we report how small an effect its sample can reliably detect, estimated by adding synthetic effects of known size to the measured per-input differences and finding the smallest that is recovered in at least 80% of simulated samples. An effect below its own detection threshold is reported as unresolved rather than as evidence of absence, and one of our own positive results, the delay rule on PG-19, is reclassified this way. An analytic cross-check, $2.83\,\hat{\sigma}/\sqrt{n}$ on the per-input differences, agrees with the simulation to within 3–5% on four rows of Table 5 and reproduces two of them exactly. It disagrees by 24% on code, whose per-input differences are close to binary on 164 inputs, so the normal approximation it rests on does not hold there, and the simulation is used throughout.

Probe sensitivity. The probe of Appendix A.10 reads a direction from a trace over several decoding steps. To check that the direction does not rest on a few extreme inputs, we recompute it after dropping the 30% of inputs that lie furthest in the observed direction. A reading whose direction holds is used as measured, and one whose direction changes is marked fragile, kept in figures as an open marker, with no claim resting on its magnitude.

The activity floor. Below an activity of 0.20, a floor chosen in advance, a null tells us nothing, because a rule that changes almost no tokens cannot show whether its changes help. The floor applies to rules that decode tokens by their own criterion, the threshold rule and the attention rule, while the delay rule defers tokens instead, so its behaviour is reported as the share of inputs on which its gate opens and the number of tokens it defers there. On open-ended text the threshold rule falls under the floor at every threshold (Figure 8), so those nulls are reported as consistent with zero with the activity printed beside them, and the open-ended conclusion rests on the delay rule, whose gate opens on 65% of C4 inputs on LLaDA and 63% on Dream and which, on the inputs where it opens, defers 45 and 43 tokens per input respectively. The attention rule is evaluated on one model and on open-ended text only, where its activity never reaches the floor at any step count, a median of 0.0091 of the masked positions at 16 steps, 0.0636 at 32 and 0.1636 at 64. Its readings are not read on the same grounds, in both directions, since at 64 steps the comparison would otherwise resolve at $+0.0022$ $[+0.0009, +0.0036]$ against a detection threshold of 0.0019. A floor that silenced nulls and admitted gains would not be a floor.

The capability floor. A cell is read only when plain decoding scores above 0.02 on arithmetic and 0.10 elsewhere, and one cell of Table 5 falls under it. Arithmetic at 16 steps has a baseline of 0.0164, twenty of 1219 questions solved, against a floor of 0.02, so that row is marked — in Table 3, its numbers are given in Table 5 under —[f], and no claim rests on it. Arithmetic at 32 and 64 steps, with baselines of 0.059 and 0.186, clears the floor and is read as before.

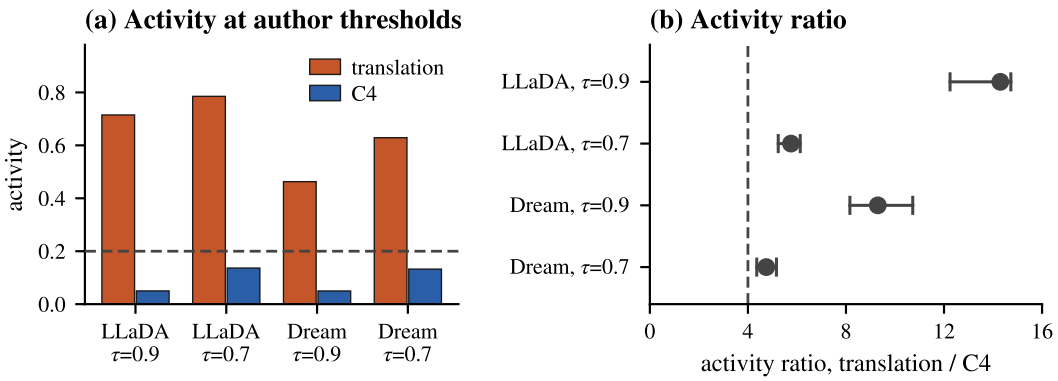


Figure 8: (a) The threshold rule's activity, the median fraction of tokens it decodes beyond the forced minimum, at both thresholds on both models, on translation and on C4. **(b)** The ratio of the two activities at 16 steps, with 95% bootstrap intervals. Dashed lines mark the activity floor of 0.20 and the ratio criterion of a lower bound of at least 4, both chosen in advance.

The copying line, and what calibrates it. The 0.02 line of §3.3 is registered on the contrast-level quantity $|(\text{copy} - \text{score})_A - (\text{copy} - \text{score})_B|$, with sign agreement as a second and independent condition, because two arms biased the same way have a known direction of contamination and two biased opposite ways do not. It was calibrated on the one translation contrast read before the control existed, whose contamination is 0.0187, the smallest line that keeps that reading. Every comparison it sets aside on magnitude alone was collected after the line was fixed; the three run before it, the threshold rule's translation contrasts on both models and Dream's delay-rule contrast at 16 steps, carry margins of opposite signs, which the sign condition sets aside at any line. Being set aside is a property of the cell and not of the model. Dream's translation comparison fails the control at 16 steps, where its two copy margins carry opposite signs and lie 0.0332 apart, and again at 64, where they are same-signed and 0.0287 apart, while at 32 steps the margins are $+0.0471$ and $+0.0368$, same-signed and 0.0102 apart, so the comparison is readable there and resolves a gain of $+0.0414$ $[+0.0202, +0.0633]$ against a detection threshold of 0.0301 on 164 items.

A.6 Scoring, guards and cost

Scorer gates. Before any collection each scorer is checked in two halves. The reference answer must score 1.0, which catches a scorer that is blind upward, and a wrong answer must score strictly lower on every item, which catches a scorer that cannot discriminate, since an average would hide a subset it cannot tell apart. The wrong answer is chosen per kind of metric. A positional metric takes the reference rotated by one token, and a set-valued metric takes the source sentence, because a rotation is invisible to a set and would score 1.0 on every item. MBPP reads 0.420 on LLaDA and 0.554 on Dream, both inside the ranges the models' own papers report.

Guards. Every number in the text is checked against the result file it comes from by a guard anchored to its section, and every table and figure is generated from those files by a script.

Cost accounting. Within every contrast the two arms spend identical forward counts per input, asserted at collection time in three parts. Across rules the step counts are not equated, since the threshold rule's cost is set by its own bar and varies per input while the delay rule runs at a fixed step count, so both conventions are reported with their own labels and never placed on one scale. Our step counts span the range of tokens per step at which published methods operate, and at one token per step our pipeline reaches 0.734 on GSM8K with LLaDA-8B whole-block, on 64 items, and 0.760 and 0.812 with the instruction-tuned models in blocks of 32.

A.7 The rules in detail

The threshold rule. Writing $c_t(i)$ for the model's confidence at masked position i, the threshold rule unmasks $C_t = \{\, i \in M_{t-1} : c_t(i) \geq \tau \,\}$ at each step, and the most confident masked position unconditionally, so that decoding always advances.

The attention rule. A third rule of our own, not read in the body because it acts on too few tokens at every step count, reuses the probe's two ingredients on the live decoding state. One attention-capturing forward gives $A_t(d)$, the median symmetrised attention between masked positions d apart for $d = 1, \ldots, 4$, and four reveals give the response size a_t as in the probe of Appendix A.10. The response profile is the attention shape normalised to that band and scaled by the response,

$$D_t(d) \;=\; \frac{A_t(d)}{\operatorname{median}_{d' \le 4} A_t(d')}\; a_t.$$

Each still-masked candidate j is then scored by how much response it would supply to everything still uncertain,

$$s_t(j) \;=\; c_t(j) \sum_{i \in M_{t-1},\, i \neq j} D_t(|i-j|)\,\big(1 - c_t(i)\big),$$

and when the gate opens the two highest-scoring candidates are added to the step's decode set. The probe runs on a copy of the state and is charged five forward passes whether or not it uses them all. Its activity, the share of positions it decodes of its own accord, is 0.009, 0.064 and 0.164 at 16, 32 and 64 steps against the floor of 0.20 of Appendix A.5, so its readings, including a $+0.0022$ $[+0.0009, +0.0036]$ on C4 at 64 steps, are printed in Table 4 and not cited.

A.8 How much of the bound a rule recovers

Appendix A.9 bounds what a rule can add above confidence ordering by $\sum_t k_t \cdot \mathrm{PCD}(C_t)$, counted in correct tokens. This section measures how much of that a rule actually recovers. The fraction recovered is the rule arm's correctly matched tokens minus its matched-action control's, over that bound, per item, on 96 items per cell. Every token count reconstructs its published score per item, extraction approximately, because its published metric is a fraction of fields and the count weights each matched field by its own length. Arithmetic is not read, because its metric is exact match, a single bit, so a token count there is a definition rather than a reconstruction and its intervals span zero at every step count.

Table 10: The bound and the fraction of it the delay rule recovers. *Bound* is the mean number of tokens per item whose most probable token changes under sequential decoding. *Recovered* is the matched-token gain over that bound, with a paired item-level interval, and *Adjacency* the same quantity restricted to positions in an adjacent pair, where measured. Arithmetic is not read for the reason in the text. Items whose bound is zero are left out by definition and counted separately.

Task	Model	T	Bound	Recovered	95% CI	Adjacency
prose	Dream	16	30.8	-0.024	$[-0.052, +0.001]$	—
prose	Dream	32	17.9	-0.009	$[-0.058, +0.037]$	—
prose	Dream	64	10.1	-0.112	$[-0.232, -0.016]$	—
prose	LLaDA	16	29.3	$+0.003$	$[-0.021, +0.029]$	-0.016
prose	LLaDA	32	17.9	$+0.027$	$[-0.017, +0.071]$	$+0.015$
prose	LLaDA	64	11.3	-0.042	$[-0.108, +0.024]$	$+0.091$
extraction	Dream	16	36.9	$+0.079$	$[+0.029, +0.134]$	$+0.003$
extraction	Dream	32	21.5	$+0.003$	$[-0.058, +0.063]$	$+0.003$
extraction	Dream	64	9.7	$+0.033$	$[-0.192, +0.226]$	—
extraction	LLaDA	16	26.3	$+0.131$	$[+0.089, +0.177]$	$+0.004$
extraction	LLaDA	32	11.1	$+0.082$	$[-0.034, +0.189]$	-0.000
extraction	LLaDA	64	5.8	$+0.270$	$[-0.052, +0.622]$	$+0.015$
arithmetic	LLaDA	16	17.8	—	—	—
arithmetic	LLaDA	32	16.8	—	—	—
arithmetic	LLaDA	64	16.5	—	—	—
translation	Dream	32	21.7	$+0.054$	$[+0.021, +0.103]$	$+0.045$
translation	LLaDA	16	18.4	$+0.044$	$[+0.022, +0.069]$	$+0.031$
translation	LLaDA	32	15.7	$+0.038$	$[+0.011, +0.067]$	—
translation	LLaDA	64	9.5	$+0.080$	$[+0.032, +0.135]$	—

Table 10 says three things. The fraction recovered is small, since where the rule helps at all it recovers between four and thirteen per cent of what the bound allows, against a bound of eighteen to thirty-seven changed tokens per item. It is smaller still on adjacency, the property the delay rule

is built around, at -0.016 to $+0.091$. And the bound is not the binding constraint on open-ended generation, where the prose cells have bounds of ten to thirty-one tokens and fractions whose intervals contain zero in five of six cells, the sixth, Dream at 64 steps, reading -0.112 $[-0.232, -0.016]$. That fraction is a share of the bound counted in matched tokens, whereas the open-ended readings of §5 are score differences against the matched-action control, and the two share neither a unit nor an interval, so the cell is reported in its own unit and is not set against §5. Between the two models we see no difference at this sample size, the adjacency fraction averaging $+0.017$ over Dream's three cells and $+0.020$ over LLaDA's seven, a difference we do not read.

A.9 Why ordering is where the value is

This section derives the bound on what any rule can add above confidence ordering by reordering or deferring within a step's decode set. It is a statement about parallel decoding, not about any method, and it is the reason the split of §6.2 has the shape it does.

Fix a step. The masked positions carry the model's marginals $p_i(\cdot)$, the data's joint is q with marginals q_i, and the schedule decodes a set C with $|C| = k$, position i receiving $\hat{t}_i = \arg\max_t p_i(t)$. Value is counted in expected correct tokens.

First, a parallel step's expected value depends on the marginals alone. By linearity, $\mathbb{E}[\#\text{correct} \mid \text{decode } C \text{ in parallel}] = \sum_{i \in C} q_i(\hat{t}_i)$, and the joint does not enter, since positions decoded at the same step are each right or wrong under their own marginal and any dependence among them affects only the variance.

Second, confidence order is optimal under those marginals. Under calibration, $p_i = q_i$, the sum $\sum_{i \in C} \max_t p_i(t)$ is largest when C is the k positions with the largest $\max_t p_i(t)$, which is confidence order, and any departure from it is worth less in expectation. Without calibration the same sentence holds with "under the model's own beliefs" in place of "optimal".

Third, dependence enters only through sequential decoding, and it is bounded by disagreement. Decode C one position at a time in confidence order and write $t_{<j}$ for the tokens decoded before j. Position j's expected value moves from $\max_t q_j(t)$ to $\max_t q_{j|t_{<j}}(t)$, and it can move either way, because the identity that puts the average of the maxima above the maximum of the average needs the earlier tokens drawn from q, whereas a greedy decoder decodes one token rather than averaging, so nothing here makes the sequential value larger. The finding of §6.3 that sequentially decoded tokens are no more often right than wrong is that other side. The difference is j's *price of parallel decoding*, and the argument needs only its size. If the most probable token is the same either way, the two decodes agree and the price is zero, and if it changes, the difference is at most one correct token in either direction. So $|\text{price}_j| \le \Pr(\arg\max_t q_{j|t_{<j}} \neq \arg\max_t q_j) =: \mathrm{PCD}_j$, and over the set

$$\mathrm{PC}(C) \;=\; \sum_{j \in C} \text{price}_j \;\le\; \sum_{j \in C} \mathrm{PCD}_j \;=\; k \cdot \mathrm{PCD}(C).$$

Any rule's gain over confidence ordering obtained by reordering or deferring inside one decode set is therefore at most $k_t \cdot \mathrm{PCD}(C_t)$ at that step. Over a decode we report $\sum_t k_t \cdot \mathrm{PCD}(C_t)$, the per-step bounds summed along the trajectory the plain decoder takes, and that sum is not itself a bound on a decode, because a rule that defers a position changes the states every later step is read at, and nothing here controls that composition. Where PCD is zero the bound is zero and no reordering can help.

What the bound estimates, and what it does not cover. Every number we report for the bound is a plug-in estimate. It reads the model's own marginals in place of the data's, on the single greedy trajectory the plain decoder takes, and no confidence interval covers the substitution. A rule that corrects the model's miscalibration gains in a way that lies outside this bound entirely. The bound counts correct tokens, so where a task's score is a monotone function of that count, as exact match, field accuracy and reference overlap are, the sign and the zero carry over to the score and the magnitude does not. We give no information-theoretic form, because the probability that a most probable token changes is not controlled in general by a total-variation distance, a near-tie changing under an arbitrarily small perturbation, and we do not state what we cannot prove.

Below eight billion parameters. No sub-1B model clears a capability floor at 16 steps. At 256 steps SMDM-1028M (Nie et al., 2025b) clears translation and arithmetic, at 0.2496 and 0.2500 on

64 items, but the probe's sign has moved by then, translation's probe interval spans zero, and its overlay comparison is set aside by the copying control, so none of the three yields a reading. The substrate's own construction on SMDM-1028M falls below the activity floor, a median fraction of 0.0818 at $\tau = 0.9$ and 0.1727 at $\tau = 0.7$, and its reading is not taken on the same grounds, which is why the substrate is stated for seven to eight billion parameters. On translation at 256 steps the attention rule copies the source more than its matched control by $+0.103$ $[+0.059, +0.149]$, an interval clear of zero, while the paired score difference spans zero, so the rule copies more and gains nothing for it, which is what the control of §3.3 exists to catch.

A.10 The uncertainty probe

Before measuring the bound we tried to build a cheap signal that would say in advance where a rule helps, and we report it because its failure is informative. The probe takes the model's partially decoded state, reveals one additional token at a fixed position on a copy of the state, and reads how much this moves the model's predictions at the positions still masked nearby. Comparing that response late in decoding with the response at the start gives a direction, negative when a revealed token moves the model less as decoding proceeds and positive when it moves it more. At 16 steps the direction separates the five tasks cleanly, negative on arithmetic, translation and extraction and positive on prose, which is also where the delay rule helps and where it does not (Figure 9).

The separation is not the task's. Read across nineteen cells, every task posed as a prompt followed by a masked block reads negative or spans zero, and every task posed as a window with scattered masks reads positive, and replacing an item's visible context with an unrelated but coherent passage leaves the direction unchanged while shuffling the context flips it in seven of the ten block-format cells (Table 11). The direction therefore tracks the format a task arrives in, which the model can see before it has generated anything. Within a task it carries no information about which inputs a rule helps, with a rank correlation of $+0.06$ $[-0.09, +0.20]$ between the probe and the delay rule's gain on GSM8K, and three predictors built from it all failed on held-out cells. What the probe measures is how much dependence the format creates, not how much of it a rule can use, and that is why the body measures the bound directly instead.

Table 11: The probe's direction at a remaining-mask fraction of 0.25 under three visible contexts and one fixed format. Entries are the range over the cells of that format; every prompt-and-block entry is negative and every scattered entry positive except where stated. Ten prompt-and-block cells (four tasks, up to three step counts) and four scattered. The extraction cells were collected under the earlier of the two extraction prompts of Appendix A.3.

Visible context	Prompt and block	Scattered
The item's own, unchanged	-2.30 to -0.31	$+0.66$ to $+0.91$
Its own tokens, shuffled	*positive in 7 of 10*	$+0.92$ to $+1.35$
Unrelated C4, coherent	-0.98 to -0.58	$+0.63$ to $+0.85$

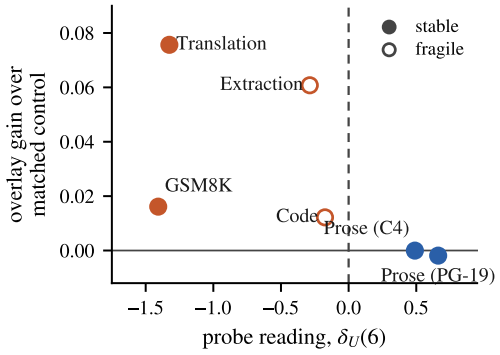


Figure 9: The delay rule's overlay against the probe reading, one point per task, at 16 steps on LLaDA-8B. Every overlay is the 16-step reading of Table 5. Filled markers survive the fragility check of Appendix A.5, open markers do not, and GSM8K's baseline sits under its capability floor at this step count, so its point carries no claim.