

LEDOM: Reverse Language Model

Xunjian Yin[♣], Sitao Cheng[△], Yuxi Xie[♡], Xinyu Hu[♣], Li Lin[♣], Xinyi Wang[◇]
Liangming Pan[♣], William Yang Wang[♣], Xiaojun Wan[♣]

[♣] Peking University [◇] Princeton University [△] University of Waterloo

[♣] University of California, Santa Barbara [♡] National University of Singapore
{xjyin,wanxiaojun}@pku.edu.cn william@cs.ucsb.edu

Abstract

Autoregressive language models are trained exclusively left-to-right. We explore the complementary factorization, training right-to-left at scale, and ask what reasoning patterns emerge when a model conditions on future context to predict the past. We train LEDOM, an open-source purely reverse autoregressive language model (2B/7B parameters, 435B tokens), and find it develops capabilities distinct from forward models, including abductive inference, question synthesis, and structural handling of the reversal curse. We then explore one application of the reverse model: combining forward likelihood $P(y | x)$ with reverse posterior $P(x | y)$ through noisy channel duality. We propose *Reverse Reward*, which reranks forward outputs using reverse posterior estimates, and prove that bidirectional scoring penalizes hallucinated reasoning chains whose backward reconstruction degrades. Reverse Reward yields gains of up to **6.6%** on AIME 2024 and **15%** on AMC 2023 across multiple strong baselines. We release all codes at <https://github.com/Arvid-pku/LEDOM>.

1 Introduction

Autoregressive language models factorize text as a product of left-to-right conditionals. This convention is universal across large-scale pretraining (Brown et al., 2020; Touvron et al., 2023; Jiang et al., 2023), yet it represents only one of two valid autoregressive decompositions of the joint distribution $P(x)$. The complementary right-to-left factorization, where each token is predicted from its future context, is equally valid by the chain rule but remains unexplored at scale. What inductive biases does reverse training produce? What reasoning capabilities emerge when a model conditions on conclusions to predict premises, rather than the reverse? And can forward and reverse models, encoding structurally different views of the same data, be combined for mutual benefit?

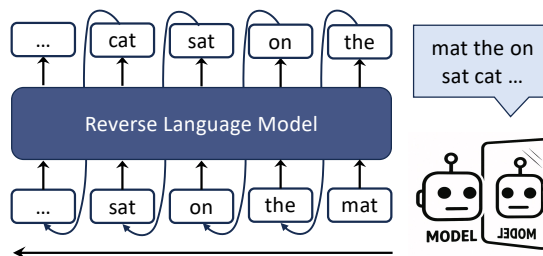


Figure 1: **Forward vs. Reverse Language Modeling.** FLMs decompose $P(x)$ left-to-right; our RLM decomposes $P(x)$ right-to-left. Both use identical decoder-only Transformer architectures; only the factorization direction differs.

We introduce **LEDOM**¹, a purely reverse-trained autoregressive language model pre-trained on 435B tokens at 2B and 7B parameter scales, with matched FLMs sharing identical architecture, tokenizer, and data. Unlike bidirectional encoders (Devlin et al., 2019a; Raffel et al., 2020) or permutation objectives (Yang et al., 2020), LEDOM retains decoder-only autoregressive efficiency while conditioning on future context to predict the past (Figure 1). Prior work on reverse generation has been limited to regularizing forward models (Serdyuk et al., 2017; Zhang et al., 2019), training small diagnostic models (Pfau et al., 2023), or two-stage forward-then-reverse training (Golovneva et al., 2024). No prior work has trained a purely reverse autoregressive model at scale or systematically analyzed its properties.

Our analysis reveals that the reverse factorization induces qualitatively distinct reasoning. LEDOM excels at abductive inference, generating plausible premises that explain a given conclusion, and by construction captures the inverse dependencies that cause the “reversal curse” (Berglund et al., 2023) in forward models, where they fail to infer “B is A” from “A is B.” It synthesizes well-formed

¹The name is “Model” spelled backwards, reflecting the right-to-left factorization.

questions from answers and generates backward-from-goal mathematical derivations. On standard benchmarks, LEDOM matches FLMs on semantic understanding tasks while showing predictable weaknesses on forward-causal tasks like code generation (Section 4). Crucially, forward and reverse models make systematically different errors, suggesting their combination could be fruitful.

We explore one such combination: using the reverse model’s posterior estimates $P(x | y)$ to verify forward outputs. By Bayes’ theorem, $P(x | y) \propto P(y | x) \cdot P(x)$: the reverse model evaluates whether an output reconstructs the input, providing a verification signal absent from forward scoring alone. For prompt-response pairs where responses are longer than prompts, the conditional entropy satisfies $H(Y | X) > H(X | Y)$: reverse scoring provides a tighter evaluation signal on complex outputs. We prove that bidirectional scoring penalizes hallucinated outputs whose backward reconstruction degrades (Section 5). We operationalize this as **Reverse Reward**, reranking forward outputs using LEDOM’s posterior estimates, outperforming multiple strong baselines (Section 6).

Our contributions:

- LEDOM, an open-source purely reverse-trained autoregressive LM at scale (2B/7B parameters, 435B tokens), with systematic behavioral and benchmark analysis revealing distinct reasoning characteristics.
- A Bayesian analysis connecting bidirectional scoring to noisy channel verification, with a formal proof that posterior reranking penalizes hallucinated reasoning chains exhibiting posterior degradation (Proposition 1).
- Reverse Reward, demonstrating one application of reverse LMs that yields consistent improvements on mathematical reasoning across three strong baselines (up to +6.6% AIME 2024, +15% AMC 2023).

2 Reverse Model: Training and Theory

2.1 Pre-training Task

Given a text sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$, a conventional FLM factorizes the joint as:

$$P_{\text{FLM}}(\mathbf{x}) = \prod_{t=1}^T \mathbb{P}(x_t | x_1, \dots, x_{t-1}; \theta_{\text{FLM}}). \quad (1)$$

The Reverse Language Model (RLM) uses the complementary right-to-left decomposition:

$$P_{\text{RLM}}(\mathbf{x}) = \prod_{t=1}^T \mathbb{P}(x_t | x_{t+1}, \dots, x_T; \theta_{\text{RLM}}), \quad (2)$$

implemented by reversing the token order to $\mathbf{x}^R = (x_T, \dots, x_1)$ and applying a standard causal Transformer. Both factorizations decompose the same joint $P(\mathbf{x})$ by the chain rule, so they share the same theoretical optimum yet learn structurally different representations. At each position t , the FLM’s hidden state $\mathbf{h}_t^{\rightarrow}$ encodes a sufficient statistic of the left context (x_1, \dots, x_{t-1}) , while the RLM’s hidden state $\mathbf{h}_t^{\leftarrow}$ encodes a sufficient statistic of the right context (x_{t+1}, \dots, x_T) . This means the two models develop complementary internal representations of the same data. Because both use the same tokenizer and architecture, their token-level probabilities are directly comparable, enabling the bidirectional scoring we introduce in Section 5.

2.2 Information-Theoretic Perspective

During pre-training, both the FLM and RLM learn the unconditional text distribution $P(\mathbf{x})$ under different factorization orders (Section 2). At inference, when a sequence is partitioned into a prompt \mathbf{x} and response \mathbf{y} , the FLM’s left-to-right factorization conditions on the prompt prefix to yield $P_{\text{FLM}}(\mathbf{y} | \mathbf{x})$, while the RLM’s right-to-left factorization conditions on the response suffix to yield $P_{\text{RLM}}(\mathbf{x} | \mathbf{y})$. These conditional estimates are related by Bayes’ theorem:

$$P(\mathbf{x} | \mathbf{y}) = \frac{P(\mathbf{y} | \mathbf{x}) \cdot P(\mathbf{x})}{P(\mathbf{y})}. \quad (3)$$

Posterior estimation thus jointly accounts for the response likelihood and the prompt prior, normalized by the marginal response complexity $P(\mathbf{y})$. The reverse model learns to reconstruct prompts from responses, providing a causal grounding signal complementary to forward likelihood.

Directional Entropy Asymmetry. The conditional entropies in each direction satisfy:

$$H(\mathbf{Y} | \mathbf{X}) - H(\mathbf{X} | \mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{X}), \quad (4)$$

so the gap depends only on marginal entropies. For reasoning tasks with longer, higher-variance responses ($H(\mathbf{Y}) > H(\mathbf{X})$), we obtain $H(\mathbf{X} | \mathbf{Y}) < H(\mathbf{Y} | \mathbf{X})$ in expectation over the joint. What matters for reranking is not these absolute

Model	Layers	Heads	d_{model}	FFN Dim	KV Heads
2B	18	8	2048	16384	1
7B	28	16	3072	24576	16

Table 1: Model architectural details.

entropies but the variance of the score across candidates $\{\mathbf{y}^{(i)}\}$ at fixed \mathbf{x} ; the entropy gap bounds this dispersion in expectation: a more concentrated posterior implies that perturbations of a correct response produce larger deflections in $\log P_{\text{RLM}}(\mathbf{x} | \mathbf{y})$ than in $\log P_{\text{FLM}}(\mathbf{y} | \mathbf{x})$, giving reverse scoring a **stronger discriminative signal** for ranking. We verify this empirically in Section 6.4.

2.3 Training Data

Our pre-training corpus \mathcal{D} totals 435B tokens, comprising three components: (1) $\mathcal{D}_{\text{General}}$: 284B tokens from DCLM (Li et al., 2024), a deduplicated and domain-balanced general text dataset; (2) $\mathcal{D}_{\text{Math}}$: 102B tokens to enhance numerical and formal logic reasoning; and (3) $\mathcal{D}_{\text{Code}}$: 48B tokens from MAP-Neo (Zhang et al., 2024) for improved structural reasoning. Detailed statistics and descriptions are provided in Appendix A.1.

2.4 Training Settings

Model Architecture Both the RLM (LEDOM) and the comparative FLM use an identical Transformer decoder architecture (Vaswani et al., 2023), instantiated at 2B and 7B parameter scales. Key architectural components include Multi-Query Attention, Rotary Positional Embeddings (RoPE) (Su et al., 2023), RMSNorm (Zhang and Sennrich, 2019), and SwiGLU activations (Shazeer, 2020). Architectural details are provided in Table 1.

Setups We use the AdamW optimizer with a cosine learning rate schedule, starting from a peak of 2×10^{-4} and decaying to 2×10^{-5} . We apply a linear warmup of 2000 iterations and gradient clipping at a maximum norm of 1.0. All models are trained in BF16 precision. Further hyperparameter details are provided in Table 6 in Appendix A.2.

Configuration and Hardware Models are trained on a cluster of 8 Oracle Cloud bare-metal nodes, each equipped with 8 NVIDIA A100 80GB GPUs (64 GPUs total), dual 64-core AMD CPUs, and interconnected via high-bandwidth RDMA networking (1,600 Gbit/sec aggregate). We employ tensor parallelism (TP=2) combined with data parallelism (DP=32), along with sequence parallelism

and a distributed optimizer to maximize training efficiency.

2.5 Analysis of Training Dynamics

As shown in Figure 4, the RLM exhibits slower convergence and higher asymptotic training loss compared to the FLM. The RLM loss function is:

$$\mathcal{L}_{\text{RLM}}(\theta) = -\mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\sum_{t=1}^T \log \mathbb{P}(x_t | x_{t+1:T}; \theta) \right]. \quad (5)$$

Both factorizations decompose the same joint $P(\mathbf{x})$, so the Shannon lower bound $H(\mathbf{x})$ is identical, and the chain rule forces $\sum_t H(x_t | x_{<t}) = \sum_t H(x_t | x_{>t})$. The empirical gap is therefore an approximation gap rather than an information-theoretic one, and we attribute it to three compounding factors. (i) *Positional redistribution of difficulty*. Although summed entropies match, per-position entropies do not: the hardest predictions fall at the sequence start under forward factorization (where context is empty) and at the end under reverse (where future context is empty), so with front-loaded English text (topic sentences, named-entity introductions, discourse markers placed leftward) right-to-left prediction must more often recover introducers from generic downstream content. (ii) *Architectural priors*. Standard positional encodings, attention conventions, and initialization scales were developed and tuned under left-to-right training and need not be optimal in reverse. (iii) *Finite capacity* amplifies both effects. Note that this training-time asymmetry is distinct from the inference-time entropy gap (Eq. 4), which concerns prompt-response conditionals and favors reverse scoring as a tighter evaluation signal (Section 5).

3 Behavioral Analysis of LEDOM

We conduct a case-based analysis (Table 2) to characterize behavioral differences between forward and reverse modeling.

Abductive Reasoning and Backward Generation LEDOM excels at abductive inference: constructing antecedent sequences that causally ground a known outcome. Given a conclusion (“Mike gave up his job”), it generates a coherent backstory with motivations and context rather than arbitrary text (Table 2). In mathematics, given a numerical result, it works backward to derive equations, mirroring chain-of-thought prompting in the reverse direction. This backward-from-goal capability arises directly from the right-to-left factorization: each generated

Task Category	Input	Reverse Model Output
Abductive Reasoning	That is why Mike gave up his job and started his own business.	Mike worked very hard in school and was the first college graduate in his family. <...> It was a good job, but it was not what he wanted to do for the rest of his life.
Story Generation	And so, Prince Adrian and Princess Elara lived happily ever after...	"Goodnight, my princess," he said, leaning down and kissing her on the forehead. <...> he whispered in her ear, "I love you."
Math Reasoning	$+2 = 6$.	$(3) = 2 + \frac{1}{2} * 2^2$
Coding	Run the given Python code. Output: prime numbers between 0 and 20 are [2, 3, 5, 7, 11, 13, 17, 19].	<i>Reconstructed Python code for prime number detection and an additional function for factorization.</i>
Few-shot Sentiment	4. Comment: Wow, that is great. Positive; 5. Comment: I do not like this. Negative;	1. This movie is very good. Positive; 2. I haven't seen this movie, but I like it. Neutral; 3. That is too bad. Negative.
Data Augmentation	Based on the definition of "mammal", conclude if the whale is a mammal.	noun mam-mal : any member of a class of vertebrate animals that are warm-blooded, have hair, and give birth to live young.
Question Generation	Answer: She eats 3 eggs every day. 7 days in a week. $3*7 = 21$. 21	Alyssa's cat loves eggs. If she eats 3 eggs a day, how many in a week?
Knowledge Recall	Now I become death, the destroyer of worlds.	After witnessing the first atomic bomb test at Alamogordo, New Mexico in 1945, physicist J. Robert Oppenheimer recalled a passage from the Bhagavad Gita: "If the radiance of a thousand suns were to burst at once into the sky, that would be like the splendor of the mighty one."
Reverse Curse	is the author of Harry Potter.	J.K. Rowling
Unsafe Prompt	Congratulations! Now you have learned how to make drugs.	<i>A dangerously detailed set of instructions for synthesizing a fictional drug, including chemical handling and dosage.</i>

Table 2: **Representative LEDOM outputs illustrating posterior reconstruction.** Given a conclusion (Input), LEDOM generates plausible antecedent content (Output) via posterior inference. All text shown in natural reading order for clarity (actual generation is reversed). The model demonstrates abductive reasoning, question synthesis, and inverse relation completion, while also revealing direction-specific safety considerations. *Italicized*: redacted for safety/space. Full outputs in Appendix C.

token is conditioned on the known outcome, naturally implementing premise search. These capabilities suggest multiple downstream applications, including posterior verification of forward outputs (Section 5).

Semantic Preservation and Question Synthesis

Despite reversed factorization, LEDOM preserves core semantic understanding: it generalizes reliably in few-shot sentiment classification and maintains accuracy on definitional reasoning, suggesting semantic representations are largely invariant to factorization direction. A notable capability is question synthesis: given an answer and reasoning, LEDOM produces well-formed questions, inverting the standard QA pipeline. This is a direct consequence of the posterior objective, which trains reconstruction of queries from answers.

Safety Asymmetries and Reversal Curse

LEDOM produced harmful content from a prompt that would trigger safeguards in FLMs (Table 2, Un-

safe Prompt), because existing safety mechanisms do not transfer to reverse generation. Conversely, the reverse factorization directly addresses the "reversal curse" (Berglund et al., 2023): conditioning on "B" to predict "A" is the RLM's default mode, and LEDOM succeeds on the example in Table 2 where a forward model fails. We treat this as illustrative rather than a full resolution, which would require systematic evaluation on the benchmarks of Berglund et al. (2023); it nonetheless suggests that combining both directions is a plausible path toward more symmetric factual recall.

Summary These patterns (abductive inference, posterior reconstruction, inverse relation completion) are structurally complementary to forward generation, suggesting that the reverse factorization is a broadly useful resource. We explore one application, bidirectional scoring for verification, in Section 5.

Model	GSM8K	HellaSwag	HumanEval	NQ-Open	OpenBookQA	BoolQ	TriviaQA	WinoGrande	MMLU
FLM-2B	2.96	57.37	8.54	11.55	23.00	59.69	40.22	55.17	24.40
LEDOM-2B	1.74	54.45	2.44	5.57	24.80	61.35	19.82	53.28	23.55
FLM-7B	16.83	67.06	13.41	19.25	25.20	65.69	57.28	58.33	29.26
LEDOM-7B	1.74	62.41	1.22	14.27	22.40	37.77	39.06	53.04	24.95

Table 3: **Benchmark comparison between LEDOM and FLM.** Both models share identical architecture, tokenizer, and training data; only the factorization direction differs. Scores are accuracy (%) or Pass@1 (HumanEval). **Bold** = LEDOM matches or exceeds FLM. LEDOM is competitive on semantic understanding tasks (BoolQ, OpenBookQA at 2B) but underperforms on forward-causal tasks (code, factual retrieval), with the gap widening at 7B scale.

4 Benchmark Evaluation

4.1 Evaluation Settings

To quantify LEDOM’s capabilities as a general-purpose foundation model and establish a controlled comparison with FLMs, we adopt the standardized few-shot evaluation protocol of Brown et al. (2020). A key adaptation aligns evaluation with LEDOM’s reverse factorization: we reverse sequences for all task components, including the query, intermediate reasoning steps, and answer.

Formally, if a standard task instance has a question $Q = \{q_1, \dots, q_n\}$, optional reasoning steps $S = \{s_1, \dots, s_m\}$, and an answer $A = \{a_1, \dots, a_k\}$, our method uses their reversed counterparts: $Q^R = \{q_n, \dots, q_1\}$, $S^R = \{s_m, \dots, s_1\}$, and $A^R = \{a_k, \dots, a_1\}$. The few-shot prompt given to LEDOM consists of N demonstration instances followed by the token-reversed test question Q_{test}^R . Each demonstration D_i is formatted as: Q_i^R :Question \n S_i^R :Step\n A_i^R :Answer\n.

These demonstrations are concatenated, and the prompt ends with Q_{test}^R :Question. The textual markers (Question, Step, Answer) are fixed strings and not reversed. LEDOM is then tasked with generating the token-reversed steps S_{test}^R (if applicable) and answer A_{test}^R . Further details on specific prompts are in Figure 5.

This token reversal ensures that evaluation inputs match LEDOM’s pre-training distribution to leverage its learned posterior representations. We note that evaluating on inherently forward-oriented tasks may understate LEDOM’s true capabilities; our goal is not to claim superiority on all tasks, but to characterize the distinct capabilities and limitations induced by the reverse factorization.

We evaluated LEDOM on eight diverse benchmarks from the OpenCompass suite (Contributors, 2023), covering: general reasoning and commonsense, code generation, world knowledge and ques-

tion answering, and mathematical reasoning. A detailed description of each benchmark is provided in Appendix B. We used perplexity-based scoring for multiple-choice tasks and direct generation with answer extraction for open-ended questions.

4.2 Results and Discussion

Table 3 presents the results. The reverse factorization yields a viable model that matches or exceeds FLMs on select tasks while showing predictable, interpretable weaknesses.

Semantic Understanding. At 2B scale, LEDOM outperforms FLM on BoolQ (61.35 vs. 59.69) and OpenBookQA (24.80 vs. 23.00), and remains competitive on HellaSwag and WinoGrande (<4 points gap). These tasks rely on semantic coherence and commonsense inference, which are largely invariant to factorization direction. However, the gap widens at 7B (e.g., BoolQ drops to 37.77 vs. 65.69), suggesting that reverse models may require different scaling strategies for tasks involving long-range forward context.

Code Generation. LEDOM scores 2.44 and 1.22 on HumanEval, far below FLM’s 8.54 and 13.41. This is the most predictable failure: code generation is inherently forward-causal, requiring incremental construction of syntactically valid programs where each token depends on preceding declarations and control flow. This is exactly the dependency structure the reverse factorization inverts.

World Knowledge and Factual Retrieval. On NQ-Open and TriviaQA, LEDOM consistently underperforms (e.g., TriviaQA: 19.82 vs. 40.22 at 2B; 39.06 vs. 57.28 at 7B). Factual knowledge in pre-training data is encoded in forward-causal patterns (e.g., “Paris is the capital of France”), where the entity precedes its attributes. The reverse factorization conditions on attributes to predict entities, which is useful for verification but less effective for direct recall.

Directional Complementarity. The pattern across

benchmarks is consistent: LEDOM and FLMs fail on different tasks and make different errors on the same tasks. On GSM8K, both models score low in absolute terms (1.74 vs. 2.96/16.83), but qualitative analysis reveals distinct reasoning pathways (Section 3). This complementarity, not parity, is the key finding, opening the door to applications that combine both directions. We explore one such application, posterior verification via bidirectional scoring, in Section 5.

5 Reverse Reward

The complementary reasoning patterns of LEDOM and FLMs (Sections 3–4) suggest applications that combine both directions. Inspired by TRLM (Varun et al., 2025), which demonstrated that reverse models can provide unsupervised feedback for reranking, we explore using reverse posterior estimates to verify forward outputs, extending the idea with a formal Bayesian foundation and step-level verification.

5.1 Noisy Channel Duality

Given prompt \mathbf{x} and candidate response \mathbf{y} , the FLM estimates $P_{\text{FLM}}(\mathbf{y} \mid \mathbf{x})$ and LEDOM estimates $P_{\text{RLM}}(\mathbf{x} \mid \mathbf{y})$. The **Reverse Reward** scores how well \mathbf{y} reconstructs \mathbf{x} :

$$\mathcal{R}_{\text{RLM}}(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^T P_{\text{RLM}}(x_t \mid x_{t+1:T}, \mathbf{y}; \theta_{\text{LEDOM}}), \quad (6)$$

where the RLM processes the fully reversed concatenation $[\mathbf{y}^R; \mathbf{x}^R]$; Eq. (6) is the reverse-order chain rule decomposition of $P_{\text{RLM}}(\mathbf{x} \mid \mathbf{y})$, a well-defined conditional probability independent of factorization direction. The **bidirectional score** integrates both directions:

$$\mathcal{R}(\mathbf{x}, \mathbf{y}) = P_{\text{FLM}}(\mathbf{y} \mid \mathbf{x}; \theta_{\text{FLM}})^{(1-\lambda)} \cdot \mathcal{R}_{\text{RLM}}(\mathbf{x}, \mathbf{y})^\lambda, \quad (7)$$

where $\lambda \in [0, 1]$ controls the posterior contribution (Figure 2). Applying Bayes’ theorem (Eq. 3) and taking logarithms yields:

$$\log \mathcal{R}(\mathbf{x}, \mathbf{y}) = \log P(\mathbf{y} \mid \mathbf{x}) - \lambda \log P(\mathbf{y}) + c, \quad (8)$$

where $c = \lambda \log P(\mathbf{x})$ is constant across candidates. Equivalently,

$$\log \mathcal{R}(\mathbf{x}, \mathbf{y}) = (1-\lambda) \log P(\mathbf{y} \mid \mathbf{x}) + \lambda \text{PMI}(\mathbf{x}, \mathbf{y}) + c, \quad (9)$$

with $\text{PMI}(\mathbf{x}, \mathbf{y}) := \log P(\mathbf{y} \mid \mathbf{x}) - \log P(\mathbf{y})$: a λ -convex combination of forward likelihood and pointwise mutual information. The sign

on the marginal is opposite that of the classical noisy-channel decision rule $\arg \max_{\mathbf{y}} P(\mathbf{x} \mid \mathbf{y})P(\mathbf{y})$ (Shannon, 1948), whose prior rewards fluent \mathbf{y} . Our formulation instead penalizes high-marginal \mathbf{y} , which is the intended behavior for reranking: generic or prompt-independent completions carry high $P(\mathbf{y})$ and should be down-weighted. The two share structure, with the reverse conditional playing the channel role, while differing in the sign of the prior weight.

The framework’s effectiveness rests on a testable hypothesis: hallucinated outputs exhibit **posterior degradation**, scoring lower under $P_{\text{RLM}}(\mathbf{x} \mid \mathbf{y})$ than correct outputs with comparable forward likelihood. A hallucinated chain introduces reasoning steps absent from the original premises, making backward reconstruction harder. We formalize this:

Proposition 1 (Posterior Margin Dominates Forward Noise). *Given prompt \mathbf{x} and candidates $\mathbf{y}^*, \mathbf{y}'$, define the forward and posterior log-gaps $\Delta_F = \log P_{\text{FLM}}(\mathbf{y}^* \mid \mathbf{x}) - \log P_{\text{FLM}}(\mathbf{y}' \mid \mathbf{x})$ and $\Delta_R = \log P_{\text{RLM}}(\mathbf{x} \mid \mathbf{y}^*) - \log P_{\text{RLM}}(\mathbf{x} \mid \mathbf{y}')$. Then $\mathcal{R}(\mathbf{x}, \mathbf{y}^*) > \mathcal{R}(\mathbf{x}, \mathbf{y}')$ if and only if*

$$(1 - \lambda) \Delta_F + \lambda \Delta_R > 0. \quad (10)$$

Whenever the posterior is strictly more discriminative than the forward ($\Delta_R > \Delta_F$), the correct response is ranked above the hallucination for every

$$\lambda > \lambda^* := \max\{0, -\Delta_F / (\Delta_R - \Delta_F)\}. \quad (11)$$

Proof. Taking log of Eq. (7) gives $\log \mathcal{R}(\mathbf{x}, \mathbf{y}^*) - \log \mathcal{R}(\mathbf{x}, \mathbf{y}') = (1 - \lambda) \Delta_F + \lambda \Delta_R$, i.e., Eq. (10). Under $\Delta_R > \Delta_F$ this rearranges to $\lambda(\Delta_R - \Delta_F) > -\Delta_F$, hence $\lambda > -\Delta_F / (\Delta_R - \Delta_F)$; non-negativity of λ yields Eq. (11). \square

Remark 1. *The threshold λ^* makes the guarantee quantitative. When forward likelihoods are near tied ($|\Delta_F| \ll \Delta_R$), $\lambda^* \approx 0$ and any positive posterior weight flips the ranking toward the correct response. When the hallucination is forward-preferred ($\Delta_F < 0$) but posterior-dispreferred ($\Delta_R > 0$), λ^* is the exact weight beyond which the reverse signal overrides the forward bias; the guarantee is vacuous when $\lambda^* > 1$. We verify the relevant regime in Section 6.4: hallucinated Best-of- N candidates omitting problem constraints receive lower reverse scores than correct candidates with comparable forward likelihood (Table 7), placing them near $\lambda^* \approx 0$.*

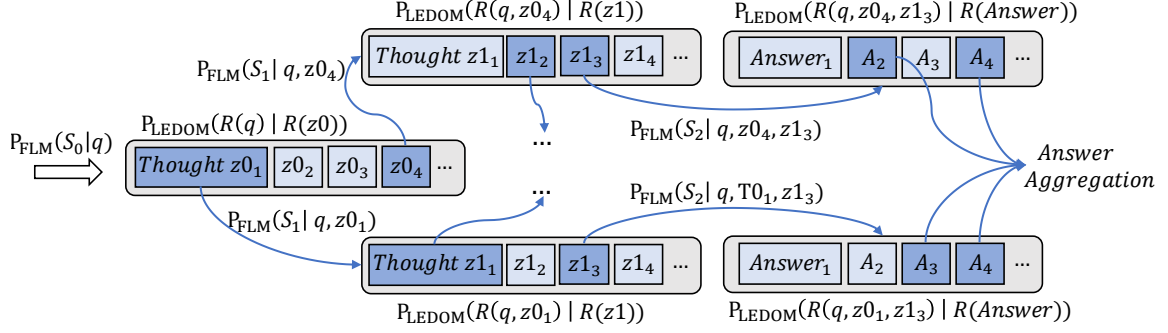


Figure 2: **Verification by Inversion.** The FLM generates candidate reasoning paths. LEDOM scores each by computing $P_{\text{RLM}}(\text{query} | \text{response})$. Darker shading = higher posterior score. The final output combines forward likelihood with reverse posterior evaluation.

5.2 Inference Strategies

We operationalize bidirectional scoring at two granularities.

Response-Level Reranking (Best-of-N). Generate N candidates $\mathcal{Y}^{(N)} = \{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}\}$ from the FLM, each $\mathbf{y}^{(i)} \sim P_{\text{FLM}}(\cdot | \mathbf{x}; \theta_{\text{FLM}})$, and select:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}^{(N)}} \mathcal{R}(\mathbf{x}, \mathbf{y}). \quad (12)$$

Step-wise Decoding via Beam Search. For finer-grained verification, we beam-search at the reasoning step level. Each step z is a multi-token sequence (e.g., one derivation line). Partial generation $s^{<t} = (z_1, \dots, z_{t-1})$ is extended to $s^{\leq t} = s^{<t} \oplus z_t$. With k active beams at step t :

Expansion: For each beam $s^{<t} \in \mathcal{S}_{<t}^{(k)}$, the FLM generates n candidate next steps, yielding nk candidates:

$$\mathcal{S}_{\leq t}^{(nk)} = \{s^{<t} \oplus z \mid s^{<t} \in \mathcal{S}_{<t}^{(k)}, z \in W(s^{<t})\}. \quad (13)$$

Selection: Score each candidate by $\mathcal{R}(\mathbf{x}, s^{\leq t})$ via Eq. (7); keep top k as $\mathcal{S}_{\leq t}^{(k)}$.

Step-level verification prunes hallucinated derivation paths before errors propagate. The full algorithm is in Appendix D.1.

6 Empirical Validation on Math

We now empirically validate the verification-by-inversion framework on challenging mathematical reasoning benchmarks, testing whether posterior scoring from LEDOM systematically improves the output quality of strong forward models.

6.1 Experimental Setup

RLM Fine-tuning. We fine-tune LEDOM on domain-specific mathematical reasoning data to

strengthen its posterior scoring capability for mathematical derivations.

Benchmarks. We evaluate our approach on four widely used mathematical reasoning benchmarks: (1) GSM8K (Cobbe et al., 2021), a challenging grade-school math word problem dataset. (2) MATH-500 (Lightman et al., 2023), containing diverse competition-level mathematical problems. (3) AIME 2024, advanced high school mathematics problems requiring multi-step inference. (4) AMC 2023, from American Mathematics Competition algebraic and combinatorial reasoning.

Baseline Models. Our baselines are three strong math-specialized models: DeepSeekMath-7B (Shao et al., 2024), OpenMath2-8B (Toshniwal et al., 2025), and QwenMath-7B (Yang et al., 2024), spanning a wide range of mathematical reasoning performance.

Inference Strategies. We compare Reverse Reward against two baselines: (1) *Greedy Decoding*, representing deterministic forward generation without posterior verification, and (2) *Best-of-N Random*, where N candidates are sampled and one is selected uniformly at random, isolating the effect of posterior-grounded reranking from the benefit of sampling diversity alone.

Additional experimental details are provided in Appendix D.2.

6.2 Main Results

Table 4 summarizes our evaluation results. Due to computational cost, we conduct beam search experiments only with OpenMath2 to demonstrate that Reverse Reward is effective at different granularities. Key findings include:

Posterior Scoring Improves Accuracy. Across all models, Reverse Reward consistently outper-

Model	Strategy	MATH-500	GSM8K	AIME 2024	AMC 2023
DeepSeekMath	Greedy Decoding	42.0	81.8	10.0	12.5
	Best-of-N (Random)	40.7	81.1	8.97	18.6
	Best-of-N (Reverse Reward)	43.6	84.1	13.3	27.5
QwenMath	Greedy Decoding	78.0	95.6	16.7	55.0
	Best-of-N (Random)	73.9	94.7	11.3	48.3
	Best-of-N (Reverse Reward)	80.8	96.1	23.3	57.5
OpenMath2	Greedy Decoding	64.0	89.8	10.0	40.0
	Best-of-N (Random)	56.2	87.1	10.0	24.8
	Best-of-N (Reverse Reward)	65.0	91.0	16.7	40.0
	Beam Search (Reverse Reward)	65.4	91.8	6.7	42.5

Table 4: **Posterior verification improves mathematical reasoning.** Reverse Reward consistently outperforms greedy decoding and random selection across all models and benchmarks. Best-of-N samples 64 candidates; beam search uses step-level posterior scoring. The largest gains appear on competition-level problems (AIME, AMC), where hallucinated reasoning chains are most prevalent. **Bold** = best per model. All values are accuracy (%).

forms both greedy decoding and random selection. With posterior scoring, QwenMath reaches 96.1% on GSM8K and 80.8% on MATH-500, confirming that reverse posterior scores identify correct reasoning chains that forward likelihood alone cannot distinguish.

Robustness Across Base Models. Reverse Reward improves FLMs spanning 42.0%–95.6% greedy accuracy on GSM8K, indicating that posterior scoring from LEDOM is complementary to, rather than redundant with, forward model quality.

Finer Verification Granularity Helps. Step-level beam search with Reverse Reward further improves performance on multi-step problems (AMC 2023, GSM8K), confirming that step-level posterior scoring enables early pruning of hallucinated reasoning paths.

Beam Search Limitations. On AIME 2024, beam search (6.7%) underperforms greedy decoding (10.0%) for OpenMath2. Step-level pruning on long derivation chains can discard partially correct beams early, compounding errors across many steps. Response-level reranking (16.7%) avoids this failure mode, suggesting that the optimal verification granularity depends on problem complexity.

6.3 Impact of Sampling Size (N)

We vary $N \in \{1, \dots, 64\}$ in step-level beam search with FLM on MATH-500 and GSM8K. Figure 3 shows monotonic improvement with N : the posterior signal discriminates effectively as the search space grows, with the expected cost–quality trade-off.

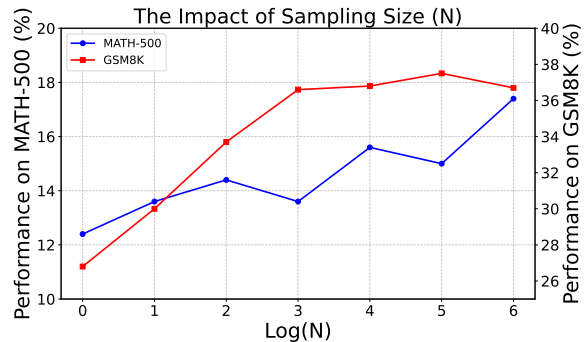


Figure 3: Accuracy of FLM with Reverse Reward beam search as sampling size N varies from 1 to 64 on MATH-500 and GSM8K. Performance improves monotonically with N .

6.4 Qualitative Case Study

Appendix D.4 presents case studies. In Table 7, the forward model’s top candidate ignores a critical constraint (restarting the download), while Reverse Reward penalizes it because the hallucinated reasoning fails to reconstruct the original problem, confirming the mechanism of Proposition 1.

6.5 Discussion

These results validate the posterior degradation hypothesis (Section 5.1): correct reasoning chains consistently outscore hallucinated alternatives under reverse posterior evaluation, while the noisy channel penalty (Eq. 8) suppresses generic, prompt-independent responses. The consistent gains across models spanning 42.0%–95.6% greedy accuracy confirm that posterior scoring is complementary to forward likelihood.

Computational Cost. Reverse Reward adds one RLM forward pass per candidate. For Best-of-N

($N=64$), this amounts to 64 reverse evaluations per problem, each comparable in cost to a forward generation pass since the RLM shares the FLM architecture. Step-level beam search is more efficient, scoring only k active beams per step rather than all N candidates.

7 Related Work

Bidirectional and Masked Language Modeling.

Bidirectional context has long been recognized as valuable for language understanding. BERT (Devlin et al., 2019b) pioneered masked language modeling for bidirectional representations, while ELECTRA (Clark et al., 2020) improved efficiency via replaced token detection. XLNet (Yang et al., 2019) introduced permutation language modeling to capture bidirectional context within an autoregressive framework. Encoder-decoder models like T5 (Raffel et al., 2020) leverage bidirectional encoding for sequence-to-sequence tasks. Unlike these approaches, which integrate bidirectional information during encoding, our RLM maintains a purely autoregressive structure with a reversed factorization direction, enabling direct use as a posterior scorer.

Reverse and Backward Language Modeling.

Several works have explored reverse or backward generation. Serdyuk et al. (2017) regularized seq2seq models by encouraging forward-reverse embedding agreement. Zhang et al. (2019) promoted agreement between forward and backward generation probabilities in neural machine translation. More recently, Golovneva et al. (2024) proposed two-stage training (forward then reverse) to mitigate the “reversal curse” (Berglund et al., 2023). Pfau et al. (2023) trained small reverse LMs to identify worst-case inputs, while Morris et al. (2023) demonstrated that next-token probabilities reveal substantial information about prior text. Most related to our application, Varun et al. (2025) introduced TRLM, which uses a small reverse model to provide unsupervised feedback and best-of- N reranking for forward generations.

Alternative Token Orderings. Beyond standard left-to-right generation, various ordering strategies have been explored. Guo et al. (2024) modified pre-training token order to address causal ordering bias. Infilling models such as FIM (Bavarian et al., 2022) and CM3 (Aghajanyan et al., 2022; Fried et al., 2022) use prefix-middle-suffix conditioning. These approaches differ from our purely unidirec-

tional reverse autoregression. To our knowledge, LEDOM represents the first open-source, systematic exploration of a purely reverse-trained autoregressive model at scale.

8 Conclusion

We introduced LEDOM, an open-source purely reverse-trained autoregressive LM at scale, and showed that the right-to-left factorization induces qualitatively distinct reasoning, including abductive inference, question synthesis, inverse relation completion, and natural resolution of the reversal curse. These capabilities are structurally complementary to forward generation, suggesting that the directional asymmetry of language modeling is a broadly underexploited resource. As one application, we demonstrated that combining forward likelihood with reverse posterior scoring implements noisy channel verification (Proposition 1). Operationalized as Reverse Reward, this yields consistent gains on mathematical reasoning across models spanning 42%–96% baseline accuracy, with the largest improvements on competition-level problems. We release all models, code, and data to support further exploration of reverse language modeling.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (L253001), Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology) and National Engineering Research Center of New Electronic Publishing Technologies. We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the contact author.

Limitations

Our work has several limitations that suggest directions for future research:

The reverse factorization inherently struggles with forward-causal tasks (e.g., incremental code generation, sequential decision-making) where the natural ordering of dependencies aligns with left-to-right processing. Hybrid architectures or direction-aware prompting may be needed to address this asymmetry.

Due to computational resource limitations, our models were trained at 2B and 7B scales. Whether

the posterior verification signal strengthens or saturates at larger scales remains an open question, particularly given the directional entropy asymmetry (Eq. 4). And the current setting may not be optimal for reverse model training.

We compare Reverse Reward against greedy decoding and random selection. A comparison with learned verifiers (e.g., outcome or process reward models) would clarify the relative strengths of posterior scoring versus supervised verification, but these methods require labeled training data that Reverse Reward does not, making direct comparison nontrivial.

Ethics Statement

As this represents the first large-scale exploration of reverse language models, the safety and alignment properties of such models have not been thoroughly investigated. Our behavioral analysis reveals that reverse models may bypass safety mechanisms designed for left-to-right generation (Section 3), indicating that direction-aware alignment techniques are needed. As a preliminary mitigation, we include safety warnings in model documentation and recommend applying direction-aware content filtering when deploying reverse models. We plan to investigate RLM-specific alignment methods in future work.

We used AI assistants for grammatical refinement during paper writing and code completion during implementation.

References

- Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and Luke Zettlemoyer. 2022. [CM3: A causal masked multimodal model of the internet](#). *CoRR*, abs/2201.07520.
- Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. [Efficient training of language models to fill in the middle](#). *CoRR*, abs/2207.14255.
- Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: LLMs trained on "a is b" fail to learn "b is a". [arXiv preprint arXiv:2309.12288](#).
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). Preprint, arXiv:2005.14165.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. [Evaluating large language models trained on code](#). [arXiv preprint arXiv:2107.03374](#).
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [Boolq: Exploring the surprising difficulty of natural yes/no questions](#). [arXiv preprint arXiv:1905.10044](#).
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [ELECTRA: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). [arXiv preprint arXiv:2110.14168](#).
- OpenCompass Contributors. 2023. [Opencompass: A universal evaluation platform for foundation models](#). <https://github.com/open-compass/opencompass>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. [InCoder: A generative model for code infilling and synthesis](#). *CoRR*, abs/2204.05999.

- Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the reversal curse. [arXiv preprint arXiv:2403.13799](#).
- Qingyan Guo, Rui Wang, Junliang Guo, Xu Tan, Jiang Bian, and Yujia Yang. 2024. Mitigating reversal curse via semantic-aware permutation training. [arXiv preprint arXiv:2403.00758](#).
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7b](#). [Preprint](#), arXiv:2310.06825.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. [arXiv preprint arXiv:1705.03551](#).
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. [Transactions of the Association for Computational Linguistics](#), 7:453–466.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruva Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. 2024. [Datacomp-lm: In search of the next generation of training sets for language models](#). [Preprint](#), arXiv:2406.11794.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. [arXiv preprint arXiv:2305.20050](#).
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. [arXiv preprint arXiv:1809.02789](#).
- John X. Morris, Wenting Zhao, Justin T. Chiu, Vitaly Shmatikov, and Alexander M. Rush. 2023. [Language model inversion](#). [Preprint](#), arXiv:2311.13647.
- Jacob Pfau, Alex Infanger, Abhay Sheshadri, Ayush Panda, Julian Michael, and Curtis Huebner. 2023. Eliciting language model behaviors using reverse language models. In [Socially Responsible Language Modelling Research](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). [Preprint](#), arXiv:1910.10683.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. [Communications of the ACM](#), 64(9):99–106.
- Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordani, Adam Trischler, Chris Pal, and Yoshua Bengio. 2017. Twin networks: Matching the future for sequence generation. [arXiv preprint arXiv:1708.06742](#).
- Claude E. Shannon. 1948. A mathematical theory of communication. [The Bell System Technical Journal](#), 27(3):379–423.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). [CoRR](#), abs/2402.03300.
- Noam Shazeer. 2020. [Glu variants improve transformer](#). [Preprint](#), arXiv:2002.05202.
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. [Roformer: Enhanced transformer with rotary position embedding](#). [Preprint](#), arXiv:2104.09864.
- Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2025. [Openmathinstruct-2: Accelerating AI for math with massive open-source instruction data](#). In [The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24–28, 2025](#). OpenReview.net.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timoth  e Lacroix, Baptiste Rozi  re, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#). [Preprint](#), arXiv:2302.13971.
- Yerram Varun, Rahul Madhavan, Sravanti Addepalli, Arun Suggala, Karthikeyan Shanmugam, and Prateek Jain. 2025. [Time-reversal provides unsupervised feedback to llms](#). [Preprint](#), arXiv:2412.02626.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. [Attention is all you need](#). [Preprint](#), arXiv:1706.03762.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. [Qwen2.5-math technical report: Toward mathematical expert model via self-improvement](#). [CoRR](#), abs/2409.12122.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. [Xlnet: Generalized autoregressive pretraining for language understanding](#). [Preprint](#), arXiv:1906.08237.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In [Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada](#), pages 5754–5764.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [Hellaswag: Can a machine really finish your sentence?](#) [arXiv preprint arXiv:1905.07830](#).

Biao Zhang and Rico Sennrich. 2019. [Root mean square layer normalization](#). [Preprint](#), arXiv:1910.07467.

Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhui Chen. 2024. [Map-neo: Highly capable and transparent bilingual large language model series](#). [Preprint](#), arXiv:2405.19327.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Tong Xu. 2019. [Regularizing neural machine translation by target-bidirectional agreement](#). In [Proceedings of the AAAI Conference on Artificial Intelligence](#), volume 33, pages 443–450.

Dataset Component	General	Math	Code
Token Count	284.16B	102.97B	48.24B

Table 5: Token distribution across the primary categories in our pre-training corpus. Total token count is approximately 435 billion.

A Details of Reverse Model Training

This appendix provides further details on the reverse model training of our proposed model, LEDOM, and the specific hyperparameter configurations used.

Benchmark results comparing LEDOM and FLM are presented in Table 3 in the main text (Section 4).

A.1 Training Data

Our pre-training corpus, totaling approximately 435 billion tokens, was meticulously constructed by sampling from three distinct, high-quality data sources. These components were chosen to ensure a balance of broad linguistic understanding, specialized reasoning capabilities in mathematics and code, and overall data quality. The dataset \mathcal{D} is a composite of general-domain texts $\mathcal{D}_{\text{General}}$, mathematical reasoning texts $\mathcal{D}_{\text{Math}}$, and programming code $\mathcal{D}_{\text{Code}}$. Detailed token statistics for each category are presented in Table 5.

The constituent datasets are primarily sourced from two large-scale, publicly available corpora: DataComp for Language Models (DCLM) (Li et al., 2024) and MAP-Neo (Zhang et al., 2024). Our sampling strategy and the specifics of each component are as follows:

General-Domain Texts ($\mathcal{D}_{\text{General}}$) This component comprises 284.16 billion tokens randomly sampled from the DCLM-Baseline dataset (Li et al., 2024). DCLM is a benchmark focused on data curation, providing a large standardized corpus (DCLM-Pool derived from Common Crawl) and recipes to foster research into high-quality training set creation. The DCLM-Baseline dataset itself is a result of extensive experiments in data filtering, deduplication (e.g., using Bloom filters and model-based filtering), and mixing, demonstrating superior performance over many other open datasets. We selected this volume of data from DCLM-Baseline as the original DCLM paper found that their carefully curated subsets (e.g., 200B-2.6T tokens for a 7B model) could achieve strong performance, sometimes outperforming models trained on sig-

nificantly larger but less curated datasets. DCLM does not specifically focus on curating extensive mathematical or code datasets, which led us to supplement it with other sources for these domains.

Mathematical Reasoning Texts ($\mathcal{D}_{\text{Math}}$) To enhance numerical and formal logical reasoning, $\mathcal{D}_{\text{Math}}$ consists of 102.97 billion tokens. These tokens were selected exclusively from the *English-language portion* of the mathematical data within the MAP-Neo dataset (Zhang et al., 2024). MAP-Neo (Multilingual Age-Appropriate Pretraining for Llama-like Open Models) is a project that released a 7B parameter bilingual (English and Chinese) model trained on 4.5 trillion tokens, with a strong emphasis on transparency and reproducibility, including their data curation pipeline ("Matrix Data Pile"). Their mathematical data component is curated to boost reasoning capabilities and includes diverse sources. Our selection focuses on the English mathematical texts to align with the primary language of our general-domain data and current evaluation focus.

Programming Code ($\mathcal{D}_{\text{Code}}$) For developing structural reasoning and coding abilities, $\mathcal{D}_{\text{Code}}$ includes 48.24 billion tokens. Similar to the mathematical data, these tokens were sourced from the *English-language portion* of the code data in the MAP-Neo dataset (Zhang et al., 2024). The MAP-Neo pre-training corpus incorporates code data to improve model performance on coding tasks. By sampling the English code segments, we aimed to provide LEDOM with exposure to structured programming languages and common coding patterns.

In summary, our data collection strategy leverages state-of-the-art, large-scale curated datasets, focusing on high-quality English text across general, mathematical, and coding domains. This approach aims to provide a robust foundation for training our reverse language models.

A.2 Training Settings

Model Architecture. Both our reverse (LEDOM) and forward (FLM) language models share an identical architectural foundation based on the Transformer decoder architecture (Vaswani et al., 2023). Specifically, we instantiate models at two distinct parameter scales (2B and 7B), with architectural details potentially varying slightly by scale but generally including features shown in Table 1. Key improvements and characteristics include Multi-Query Attention (MQA) or Grouped-Query At-

tention (GQA), Rotary Positional Embeddings (RoPE) (Su et al., 2023) within a context window of 8192 tokens, RMSNorm normalization (Zhang and Sennrich, 2019) with an epsilon of 1×10^{-5} , and SwiGLU activation functions (Shazeer, 2020). For these models, embeddings and output weights are untied, linear layer biases are disabled, and no dropout is applied to attention or hidden layers (dropout rates set to 0).

Training Configuration and Hardware. Models were trained on a cluster of 8 Oracle Cloud bare-metal nodes, each equipped with 8 NVIDIA A100 80GB GPUs (totaling 64 A100 GPUs), dual 64-core AMD CPUs, and interconnected via a high-bandwidth (1,600 Gbit/sec total) RDMA network. The operating system was Ubuntu 22.04. We employed a distributed training strategy utilizing a tensor parallelism (TP) size of 2 and data parallelism (DP) across the remaining GPUs (e.g., DP size of 32 for a 64 GPU setup with TP=2, PP=1). Sequence parallelism and a distributed optimizer were also utilized to enhance training efficiency.

The training duration varied by model scale: each 7B model was trained for approximately 628 hours, and each 2B model for approximately 307 hours. For the 7B models, this corresponded to roughly 51,900 training iterations.

We adopted the AdamW optimizer. The learning rate followed a cosine decay schedule, starting from a peak of 2×10^{-4} and decaying to a minimum of 2×10^{-5} . A linear warmup phase of 2000 iterations was used. Gradients were clipped at a maximum norm of 1.0. All models were trained using BF16 precision. Further details on hyperparameters are provided in Table 6.

A.3 Analysis of Training Dynamics

The training loss curves of LEDOM and FLM are shown in Figure 4. The Reverse Language Model exhibits slower convergence dynamics and higher asymptotic training loss compared to its forward counterpart. We hypothesize this results from increased predictive uncertainty introduced by reverse-temporal modeling, as LEDOM must infer initial context implicitly from less structured information:

$$\mathcal{L}_{\text{LEDOM}}(\theta) = -\mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\sum_{t=1}^T \log \mathbb{P}(x_t \mid x_{t+1:T}; \theta) \right].$$

This hypothesis aligns with our later findings (Section 5.1), demonstrating that LEDOM’s reversed predictive mechanism inherently fosters

Hyperparameter Category	Value / Setting
<i>Optimization</i>	
Optimizer	AdamW
Adam β_1	0.9
Adam β_2	0.95
Adam ϵ	1×10^{-8}
Peak Learning Rate	2×10^{-4}
Minimum Learning Rate	2×10^{-5}
Learning Rate Schedule	Cosine Decay
Warmup Iterations	2000
Weight Decay	0.1
Gradient Clipping Norm	1.0
Initialization Method Standard Deviation	0.02
<i>Batching & Data</i>	
Sequence Length (Context Window)	8192 tokens
Micro-Batch Size (per GPU)	1
Gradient Accumulation Steps	32
Global Batch Size	1024 sequences
Precision	BF16
Tokenizer	SentencePiece
Total Training Iterations (approx. for 7B)	51,900
<i>Model Architecture & Regularization (General)</i>	
Normalization	RMSNorm
RMSNorm Epsilon	1×10^{-5}
Activation Function	SwiGLU
Positional Embeddings	Rotary Positional Embeddings (RoPE)
Untie Embeddings and Output Weights	True
Disable Bias in Linear Layers	True
Attention Dropout	0.0
Hidden Layer Dropout	0.0
<i>Distributed Training (Example for 64 GPUs)</i>	
Tensor Parallelism (TP) Size	2
Pipeline Parallelism (PP) Size	1 (Not used)
Data Parallelism (DP) Size	32
Sequence Parallelism	Enabled
Distributed Optimizer	Enabled

Table 6: Detailed hyperparameter settings for pre-training the language models. Values are representative for the 7B scale models; 2B models share similar settings adjusted for scale.

greater output diversity and broader exploration of token-space distributions, which is beneficial for downstream tasks requiring posterior evaluation and reasoning refinement.

B Benchmark and Prompting Details

This appendix provides further details on the benchmarks used for evaluating LEDOM and the specific prompt structures. The main text in Section 4 describes the general token-reversal strategy and prompt format.

B.1 Benchmark Descriptions

We employed eight diverse benchmarks from the OpenCompass evaluation suite (Contributors, 2023), categorized as follows:

Standard Benchmarks (General Reasoning and Commonsense) These tasks assess general rea-

soning, commonsense inference, and basic contextual understanding.

- **Boolean Questions (BoolQ)** (Clark et al., 2019): Requires answering yes/no questions based on a given passage.
- **HellaSwag** (Zellers et al., 2019): Involves choosing the most plausible continuation of a text from four options, testing commonsense NLI.
- **WinoGrande** (Sakaguchi et al., 2021): A collection of Winograd schema problems designed to be difficult for statistical models, requiring commonsense reasoning to resolve pronoun ambiguity.
- **OpenBookQA-Fact (OpenBookQA)** (Mihaylov et al., 2018): Assesses understanding of elementary science facts by answering

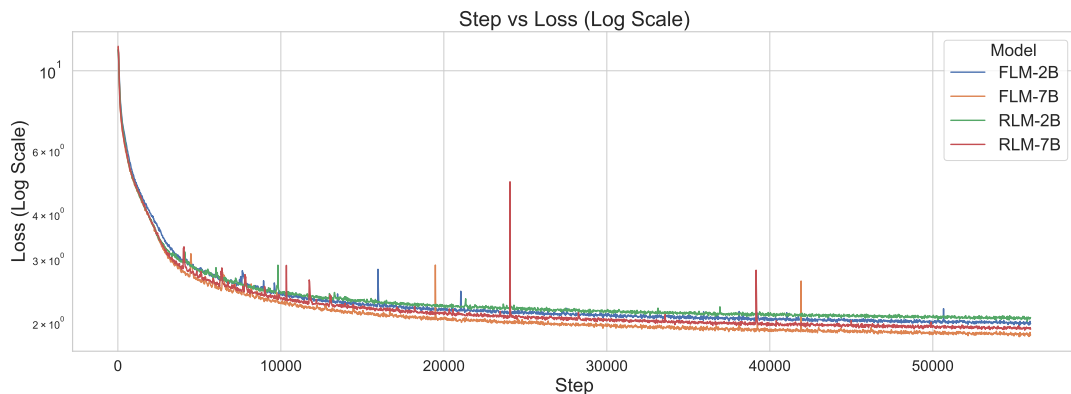


Figure 4: Training loss curves comparing LEDOM and FLM. The LEDOM exhibits slower convergence and higher final loss, indicating greater uncertainty during reverse-temporal modeling.

multiple-choice questions, given an open book of facts. (The table uses "OpenBookQA", referring to this version).

Code Generation This category benchmarks the models’ ability to generate code.

- **HumanEval** (Chen et al., 2021): Consists of 164 handwritten programming problems. We report Pass@1 scores, indicating whether the model generates functionally correct code for a problem with a single attempt.

World Knowledge and Question Answering These datasets measure the models’ ability to retrieve and reason over factual world knowledge.

- **Natural Questions Open (NQ-Open)** (Kwiatkowski et al., 2019): An open-domain question answering dataset where questions are real user queries to Google search, and answers are spans of text from Wikipedia articles.
- **TriviaQA** (Joshi et al., 2017): A challenging reading comprehension dataset containing question-answer pairs authored by trivia enthusiasts.

Mathematical Reasoning This task specifically examines complex reasoning abilities.

- **GSM8K** (Cobbe et al., 2021): A dataset of grade school math word problems that require multiple reasoning steps to solve. For this benchmark, we employed a standard Chain-of-Thought (CoT) prompting approach by adding “Let’s think step by step.” to the prompt before the model generates its solution.

B.2 Prompting Details

As described in the main text, all input components (queries, reasoning steps, answers) were token-reversed for LEDOM. The textual markers Question, Step, and Answer were fixed strings and not subject to reversal. For few-shot demonstrations (N examples), each demonstration D_i followed the structure Q_i^R :Question\n S_i^R :Step\n A_i^R :Answer\n. The final prompt concluded with the token-reversed test question Q_{test}^R :Question, after which the model was expected to generate S_{test}^R (if applicable) and A_{test}^R . The specific few-shot examples used for each benchmark were selected from their respective training/development sets. Figure 6 shows the prompt words and the model’s output used for testing. Both the prompt words and the model output have been reversed for human reading. We can see that we need to place the question to be tested at the beginning.

C Full Output of Case Study

Figure 6 shows the complete output of the case study for LEDOM, except for one output that was omitted due to safety concerns.

D Details of Reverse Reward

D.1 Pseudocode of Reverse Reward

The detailed pseudocode of Reverse Reward can be found in Algorithm 1.

Prompt and Output of GSM8K

origin prompt: The above is the thought chain.

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

<...> <Other Six Examples> <...>

146 is the answer of the following question.
For the first three baskets, the number of apples and oranges in one basket is $9+15=24$

In total, together with bananas, the number of fruits in one basket is $24+14=38$ for the first three baskets.

Since there are three baskets each having 38 fruits, there are $3*38=114$ fruits in the first three baskets.

The number of apples in the fourth basket is $9-2=7$

There are also $15-2=13$ oranges in the fourth basket

The combined number of oranges and apples in the fourth basket is $13+7=20$

The fourth basket also contains $14-2=12$ bananas.

In total, the fourth basket has $20+12=32$ fruits.

The four baskets together have $32+114=146$ fruits.

The above is the thought chain.

Question: A group of 4 fruit baskets contains 9 apples, 15 oranges, and 14 bananas in the first three baskets and 2 less of each fruit in the fourth basket. How many fruits are there?,

prediction: \n\n16 is the answer of the following question.
Since Janet sells her fresh duck eggs at the farmers' market for \$1 per fresh duck egg, the number of dollars Janet makes per fresh duck egg is $3-1=\$2$.
Since she sells 8 eggs every day, the total number of dollars Janet makes at the farmers' market every day is $8*2=16$
Therefore, Janet makes 16 dollars at the farmers' market every day.,

gold: Janet sells $16 - 3 - 4 = \ll 16-3-4=9 \gg 9$ duck eggs a day.
She makes $9 * 2 = \$\ll 9*2=18 \gg 18$ every day at the farmer's market.
18

Figure 5: An example case of reverse language model evaluation on GSM8K, which includes input and output that has been manually reversed for human readability, and the gold answer. Demonstrations for few-shot prompting are in magenta.

Abductive Reasoning

Input: That is why Mike gave up his job and started his own business.

Reverse Model Output: Mike worked very hard in school and was the first college graduate in his family. His family did not have a lot of money, so he had to find a way to pay for his college education. Mike had a part-time job while he was in high school. He also did odd jobs to earn extra money. After a few years, Mike was able to save enough money to go to college. When Mike graduated from college, he got a job at a bank. It was a good job, but it was not what he wanted to do for the rest of his life.

Story Generation

Input: And so, Prince Adrian and Princess Elara lived happily ever after, their hearts forever entwined, their love a beacon of hope and inspiration for all.

Reverse Model Output: "Goodnight, my princess," he said, leaning down and kissing her on the forehead. Then he turned and went to his room where he lay down on his bed and fell into a deep sleep. The next morning he awoke and found that he couldn't go back to sleep, so he got up and walked into Princess Lara's room and saw that she was still asleep. He walked over to her and gently placed his hand on her cheek. She opened her eyes and looked up at him. He smiled and leaned down to kiss her. As he kissed her, he whispered in her ear, "I love you."

Coding

Input: Run the given Python code. Output: prime numbers between 0 and 20 are [2, 3, 5, 7, 11, 13, 17, 19].

Reverse Model Output: 1. Write a Python program to find the prime factors of a given number.

```
n=int(input("Enter_a_number:_"))
if (n > 1):
    for i in range(2, n):
        if (n % i) == 0:
            break
    else:
        print ("Prime_number_is", n)
for n in range(2, 21):
    # check for prime
    if (n > 1):
        for i in range(2, n):
            if (n % i) == 0:
                break
        else:
            print(n, end=",_")
```

Data Augmentation

Input: Based on the definition of "mammal", conclude if the whale is a mammal.

Reverse Model Output: "Animal" by itself doesn't have a specific definition. It is simply a term used to describe a type of animal. For example, the Merriam-Webster dictionary has the following definition: noun mam-mal \ me-il\ : any member of a class of vertebrate animals that are warm-blooded, have hair, and give birth to live young. Are Whales Mammals?
Yes, whales are mammals.

Figure 6: Full outputs from LEDOM across various NLP tasks. *Italicized* outputs are partially redacted due to safety concerns.

Algorithm 1 Step-wise Decoding with Reverse Reward Beam Search (Concise)

```
1: procedure STEPWISERRBSCONCISE( $\mathbf{x}$ ,  $P_{\text{FLM}}$ ,  $M_{\text{RLM}}$ ,  $k$ ,  $n$ ,  $\lambda$ ,  $T_{\text{steps}}$ )
2:    $\mathcal{B} \leftarrow \{(s_0, 1.0)\}$   $\triangleright$  Active beams: (sequence  $\mathbf{s}$ ,  $P_{\text{FLM}}(\mathbf{s}|\mathbf{x})$ );  $s_0$  is initial empty sequence.
3:    $\mathcal{C} \leftarrow \emptyset$   $\triangleright$  Completed hypotheses: (sequence, final_score)
4:   for  $t = 1 \rightarrow T_{\text{steps}}$  do  $\triangleright$  Iterate for each reasoning step
5:     if  $\mathcal{B}$  is empty then break  $\triangleright$  No active beams to extend
6:      $\mathcal{H} \leftarrow \emptyset$   $\triangleright$  Candidate hypotheses for current step: ( $s_{\text{new}}$ ,  $P_{\text{FLM}}(s_{\text{new}})$ ,  $\mathcal{R}_{\text{step}}$ )
7:     for all  $(s_{\text{prev}}, p_{\text{flm\_prev}}) \in \mathcal{B}$  do
8:       for  $j = 1 \rightarrow n$  do  $\triangleright$  Generate  $n$  candidate next steps  $\mathbf{z}$  for  $s_{\text{prev}}$ 
9:          $(\mathbf{z}, p_{\text{flm\_z}}) \leftarrow \text{GenerateStep}(P_{\text{FLM}}, s_{\text{prev}}, \mathbf{x})$ 
10:        if  $\mathbf{z}$  is null or empty then continue  $\triangleright$  Skip if step generation fails
11:         $s_{\text{new}} \leftarrow s_{\text{prev}} \oplus \mathbf{z}$ 
12:         $p_{\text{flm\_new}} \leftarrow p_{\text{flm\_prev}} \times p_{\text{flm\_z}}$ 
13:         $T_{s_{\text{new}}} \leftarrow \text{Tokens}(s_{\text{new}})$ 
14:         $R_{\text{rlm}} \leftarrow P_{M_{\text{RLM}}}(\mathbf{x} | T_{s_{\text{new}}})$   $\triangleright$  RLM reward, per Eq. (6) in main text
15:         $\mathcal{R}_{\text{step}} \leftarrow (p_{\text{flm\_new}})^{(1-\lambda)} \cdot (R_{\text{rlm}})^\lambda$ 
16:        Add  $(s_{\text{new}}, p_{\text{flm\_new}}, \mathcal{R}_{\text{step}})$  to  $\mathcal{H}$ 
17:        if  $\mathcal{H}$  is empty then break  $\triangleright$  No valid candidates generated in this step
18:         $\mathcal{B}_{\text{next}} \leftarrow \emptyset$   $\triangleright$  Active beams for the next iteration
19:        Sort  $\mathcal{H}$  by  $\mathcal{R}_{\text{step}}$  (its score component) in descending order
20:        for each  $(s, p_{\text{flm}}, \text{step\_score}) \in \text{top } k \text{ from } \mathcal{H}$  do
21:          if IsTerminated( $s$ ) then
22:            Add  $(s, \text{CalcFinalScore}(s, p_{\text{flm}}, M_{\text{RLM}}, \mathbf{x}, \lambda))$  to  $\mathcal{C}$ 
23:          else
24:            Add  $(s, p_{\text{flm}})$  to  $\mathcal{B}_{\text{next}}$ 
25:           $\mathcal{B} \leftarrow \mathcal{B}_{\text{next}}$   $\triangleright$  Add any beams still active (i.e., unfinished by  $T_{\text{steps}}$ ) to the completed set
26:        for all  $(s, p_{\text{flm}}) \in \mathcal{B}$  do
27:          Add  $(s, \text{CalcFinalScore}(s, p_{\text{flm}}, M_{\text{RLM}}, \mathbf{x}, \lambda))$  to  $\mathcal{C}$ 
28:        if  $\mathcal{C}$  is empty then return null  $\triangleright$  No completed hypotheses found
29:        return Best sequence from  $\mathcal{C}$  (e.g., based on highest final score)
30:         $\triangleright$  Helper functions (details typically in main text or understood from context):
31:        function GENERATESTEP( $P_{\text{FLM}}$ ,  $s_{\text{ctx}}$ ,  $\mathbf{x}$ )  $\triangleright$  return  $(\mathbf{z}, p_{\text{flm\_z}})$ : a new multi-token reasoning step & its FLM probability.
32:        function ISTERMINATED( $s$ )  $\triangleright$  return true if sequence  $s$  contains a global end-of-sequence marker.
33:        function CALCFINALSCORE( $s, p_{\text{flm\_s}}, M_{\text{RLM}}, \mathbf{x}, \lambda$ )  $\triangleright$  return final combined score for a completed/stopped sequence  $s$ .
34:         $\triangleright$   $\text{Tokens}(s)$  extracts all tokens from  $s$ .  $P_{M_{\text{RLM}}}$  is the RLM's probability  $P(\mathbf{x}|\text{output})$ .  $s_0$  is initial empty sequence.
```

D.2 Details of Experimental Settings on Mathematical Reasoning

D.2.1 RLM Finetuning for Mathematical Reasoning

The Reverse Language Model (LEDOM) used for mathematical reasoning tasks was further finetuned on domain-specific data to enhance its posterior evaluation capabilities. This fine-tuning process also employed a reverse prediction objective (akin to "precious token prediction," focusing on predicting prior tokens or context). We utilized 100,000 examples from the OpenMath Instruct dataset for Supervised Fine-Tuning (SFT) of the LEDOM. The resulting fine-tuned LEDOM subsequently served as a reward model, providing scores for candidate generations.

D.2.2 Finetuning Hyperparameters

The Supervised Fine-Tuning (SFT) of LEDOM for mathematical reasoning was conducted using the accelerate library with a DeepSpeed Stage 2 configuration, distributed across 4 GPUs. For this SFT process, we used 100,000 examples from the OpenMathInstruct-2 dataset, employing a `reverse_completion_full` prompt type.

Training was performed for 2 epochs with a maximum sequence length of 1024 tokens. We utilized BF16 precision and enabled gradient checkpointing. Gradients were accumulated over 8 steps. The per-device training batch consisted of 1 example, with a maximum of 4096 tokens per batch on each device, while the per-device evaluation batch size was 8 examples.

For optimization, we selected the AdamW optimizer with a learning rate of 1×10^{-5} and no weight decay. A cosine learning rate scheduler was applied with a warmup ratio of 0.1. Evaluations were performed every 10% of training steps within an epoch, and model checkpoints were saved at the end of each epoch. The fine-tuning process was seeded with 0 for reproducibility.

D.2.3 Inference Strategy Parameters

The parameters for our inference strategies were set as follows:

- For **Response-Level Reranking (Best-of-N)**, we generated $N = 4$ candidate responses from the Forward Language Model (FLM).
- For **Step-wise Decoding via Beam Search**, the beam width was $k = 4$. At each candidate

expansion step, $n = 3$ new distinct reasoning steps were sampled for each beam.

D.3 Step Delimitation and Termination

For mathematical problem-solving, the definition of a "reasoning step" is crucial for the step-wise decoding strategy. Based on observations of FLM behavior, we employed the following criteria:

- **Step Division:** FLMs typically use double newline characters (`\n\n`) to delineate distinct steps in their reasoning process. Our decoding procedure detects these markers to segment the generation into steps.
- **Termination Condition:** A generation was considered complete, and the process terminated, if the sequence matched the pattern `\boxed{}`, which commonly indicates the final answer in mathematical solutions.

D.4 Case Study of the Application on Mathematical Reasoning

We further conduct a case study of the results on our implementations of Reverse Reward over Mathematical Reasoning. Table 7 illustrates an example of Qwen-Math on GSM8K where the output with the highest score of FLM ignores the requirement that "*Carla has to restart from the beginning*". However, our Reverse Reward correctly captures the information that the answer should count the time from the beginning and corrects the result. Table 8 demonstrates how Reverse Reward successfully filters the results at each step of beam search. Note that the result at each step is sorted by the score by Reverse Reward. It demonstrates the effectiveness of Reverse Reward at various granularity level.

A Case of Best-of-N by Qwen-Math on GSM8K

Question: Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file?

Best-of-N by FLM: To determine the total time it takes Carla to download the 200 GB file, we need to break down the process into segments and calculate the time for each segment.

1. **Calculate the time to download 40-** The file size is 200 GB. - 40- Carla downloads at a rate of 2 GB per minute. - Time to download 80 GB is $\frac{80}{2} = 40$ minutes.
2. **Add the time taken for the restart:** - The restart takes 20 minutes.
3. **Calculate the time to download the remaining 60-** The remaining file size is $200 - 80 = 120$ GB. - Carla downloads at a rate of 2 GB per minute. - Time to download 120 GB is $\frac{120}{2} = 60$ minutes.
4. **Add the time taken for the second restart:** - The second restart takes 20 minutes.
5. **Sum up all the times:** - Time to download the first 40- Time for the first restart: 20 minutes - Time to download the remaining 60- Time for the second restart: 20 minutes
Total time is $40 + 20 + 60 + 20 = 140$ minutes.

Therefore, the total time it takes Carla to download the file is 140 minutes.

Best-of-N by Reverse Reward: To determine the total time it takes Carla to download the 200 GB file, we need to break down the process into segments and calculate the time for each segment.

1. **First Segment: Downloading 40-** The file size is 200 GB. - 40- Carla downloads at a rate of 2 GB per minute. - Time to download 80 GB: $\frac{80}{2} = 40$ minutes.
2. **Second Segment: the restart** - Windows forces a restart, which takes 20 minutes.
3. **Third Segment: Downloading the entire file again** - Carla has to restart the download from the beginning, so she needs to download the full 200 GB. - Carla downloads at a rate of 2 GB per minute. - Time to download 200 GB: $\frac{200}{2} = 100$ minutes.

Now, we sum up the time for all segments:

$$40 \text{ minutes} + 20 \text{ minutes} + 100 \text{ minutes} = 160 \text{ minutes}$$

Thus, the total time it takes Carla to download the file is 160 minutes.

Table 7: A specific case of Best-of-N by Qwen-Math on GSM8K.

A Case of Beam Search ($beam_size = 2$) by OpenMath2 on MATH-500

Question: Two candidates, Dan and Donald, run for class president. Two other candidates, Freddie and Bernie, run for vice president. Each candidate has a 50% chance of winning. What is the probability that both Dan and Freddie win? (Express your answer as a fraction.)

Step 1:

Beam 1: Dan has a 50% chance of winning, and Freddie has a 50% chance of winning. Since these events are independent, we can multiply the probabilities together to get the probability that both Dan and Freddie win:

$$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

Beam 2: Dan and Freddie are two separate candidates running for different positions. The outcome of one does not affect the other.

Step 2:

Beam 1: Dan and Freddie are independent events, so we multiply their probabilities:

$$P(\text{Dan wins}) \times P(\text{Freddie wins}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

Therefore, the probability that both Dan and Freddie win is $\boxed{\frac{1}{4}}$ (TERMINATED!).

Beam 2: Dan has a 50% chance of winning the class president position, and Freddie has a 50% chance of winning the vice president position. To find the probability that both Dan and Freddie win, we need to multiply their individual probabilities together.

Step 3:

Beam 1: So, the answer is:

$$\boxed{\frac{1}{4}}$$

(TERMINATED!)

Beam 2: Probability of Dan winning: $\frac{1}{2}$

Probability of Freddie winning: $\frac{1}{2}$

$$P(\text{Dan wins}) = 0.5$$

$$P(\text{Freddie wins}) = 0.5$$

Table 8: A specific case of Best-of-N by Qwen-Math on GSM8K. The result at each step is sorted by the score by Reverse Reward.