REINFORCED GENERATIVE RECURSIVE REASONING FOR NATIVE ADAPTIVE COMPUTATION DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive Reasoning Models scale computation by iteratively refining a latent state with shared transition functions, yet the amount of recursion is usually fixed externally or predicted by an auxiliary head. We show how a generative recursive prior turns this limitation into a learnable decision: its Gaussian transition is a normalized, reparameterizable policy with an exact log-density. Variational pretraining therefore provides an on-manifold starting distribution, while outcome reinforcement learning can update both latent transitions and a residual-conditioned halting action. A potential-based value signal supplies credit across the loop and a KL anchor keeps exploration near the pretrained manifold. On structured reasoning benchmarks, ReGRAM improves ARC-AGI-1/2 accuracy by 4.8/5.7 percentage points over GRAM, allocates more depth to harder inputs, and transfers from a training depth cap of 16 to inference budgets of 1024. The approach motivates a simple principle: adaptive depth is most natural when the computation itself is represented as a stochastic policy.

1 INTRODUCTION

A central question for neural reasoning systems is not only *how* to perform extended computation, but *how much* of it to perform. Autoregressive reasoners answer this question implicitly: a chain-of-thought ends when the model emits a stop token, and the length of the thought is therefore a decision the model learns to make (Wei et al., 2022; Guo et al., 2025). The breakthrough of reinforcement-learned reasoning—most visibly in DeepSeek-R1 (Guo et al., 2025)—is precisely that a single outcome reward, with no supervision on length, causes long deliberate reasoning chains to *emerge*: the model discovers on its own that harder problems are worth more tokens. Length is not prescribed; it is shaped by reward.

Recursive Reasoning Models (RRMs) repeatedly apply a shared transition function to refine a persistent latent state (Yang et al., 2024; Wang et al., 2025; Jolicoeur-Martineau, 2025; Dehghani et al., 2019), decoupling reasoning depth from parameter count and output length. Small task-specific recursive models solve Sudoku and ARC without internet-scale pretraining (Wang et al., 2025; Jolicoeur-Martineau, 2025); the ARC leaderboard provides broader context across model scales and inference budgets (ARC Prize Foundation, 2026). These models typically train under a prescribed recursion budget and use auxiliary mechanisms to allocate computation: ACT heads (Dehghani et al., 2019; Wang et al., 2025), Q-learning stop classifiers (Wang et al., 2025; Jolicoeur-Martineau, 2025), or residual thresholds (Huang et al., 2026). Depth-recurrent language models also support adaptive inference (Geiping et al., 2025). ReGRAM builds on this line of work by jointly optimizing a stochastic recursive transition and its stopping rule from the same verifiable reward.

Native reward-driven depth is difficult because a likelihood-ratio estimator requires a normalized distribution over the decisions being changed. Autoregressive reasoning exposes this interface for every token, including the stop token. A deterministic recursive core instead exposes a point transition, so outcome credit cannot reach the transition through a score term unless one adds a stochastic proposal or a separate surrogate. ACT and Q-head methods can learn useful stopping rules, but their stopping objective is typically decoupled from the transition update.

GRAM (Baek et al., 2026) supplies the missing interface. Each recursive update is sampled from a learned Gaussian transition prior, so the transition is a stochastic policy with a tractable log-density whose parameters are the reasoning weights. This raises a focused question: when a recursive model

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.

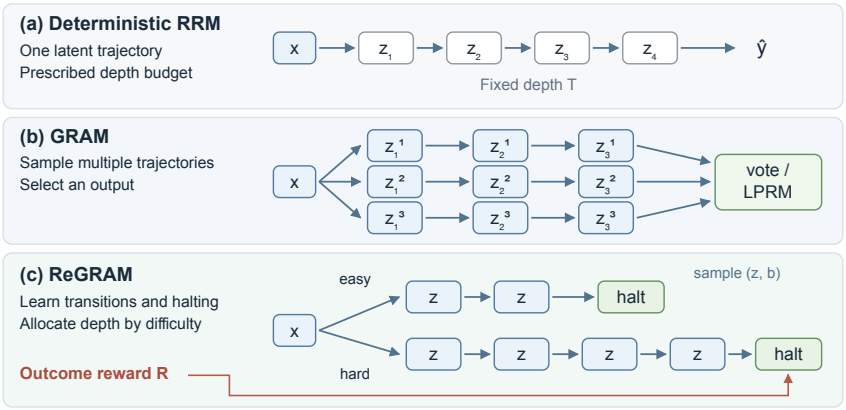


Figure 1: From prescribed to learned depth. (a) A deterministic recursive core follows one latent trajectory under a prescribed depth budget; auxiliary heads can predict halting. **(b)** GRAM samples multiple trajectories and selects their outputs by voting or an LPRM under a fixed budget. **(c)** ReGRAM jointly learns stochastic transitions and residual-conditioned halting from outcome reward, allocating shorter trajectories to easy instances and longer trajectories to hard ones.

already defines a normalized trajectory distribution, can variational pretraining and policy-gradient post-training optimize that same distribution without an empirical surrogate? The Gaussian prior provides both the required likelihood and a natural trust-region reference. Prior work interprets deterministic TRM recursion as a policy-improvement operator (Asadulaev et al., 2025), while recent latent reasoners construct surrogate policies for outcome-RL and learned stopping (You et al., 2026); our setting keeps the deployable transition distribution itself in the loop.

The proposed extension adds a residual-conditioned Bernoulli halt action to GRAM and post-trains the joint transition-and-halt policy with a hybrid score/pathwise update. Exact Gaussian likelihoods and alive-at-step group baselines assign verifier credit to latent transitions; a value potential spreads the terminal signal across the loop; and a KL anchor keeps the policy on the pretrained latent manifold. Residual clipping makes the halting input bounded under a contractive-basin assumption. RecurTrace (Wang et al., 2026) provides a complementary deterministic oracle-distilled baseline, whereas our claim concerns reward-driven joint learning.

Our contributions are: (1) We identify the normalized generative transition as the representation that makes native adaptive depth amenable to outcome-RL. (2) We give a two-phase recipe that reuses this distribution for variational pretraining and policy-gradient post-training, with a residual-conditioned halt action that can transfer beyond the training horizon. (3) We demonstrate accuracy gains, difficulty-dependent depth, and transfer beyond the training horizon on structured reasoning tasks, and quantify the correctness–coverage tradeoff on multi-solution constraints.

2 Background: Generative Recursive Reasoning

We build on GRAM (Baek et al., 2026) and use its notation for stochastic transitions, variational training, and inference-time sampling.

Stochastic latent transitions. GRAM models a conditional distribution $p_\theta(y \mid x)$ by marginalizing over latent reasoning trajectories. Given an input x, an encoder produces a conditioning embedding $e_x = f_{\text{enc}}(x;\theta)$ that is reused throughout. Starting from a fixed initial state z_0, the model evolves the latent state through learned *stochastic* transitions

$$z_t \sim p_\theta(z_t \mid z_{t-1}, e_x), \qquad t = 1, \dots, T, \tag{1}$$

and decodes a prediction $\hat{y} = \arg\max f_{\text{dec}}(z_T;\theta)$ from the terminal state. The transition is realized as a learned stochastic residual around a deterministic update: the model computes a deterministic proposal u_t, samples a state-dependent Gaussian perturbation, and adds it,

$$\epsilon_t \sim p_\theta(\epsilon_t \mid u_t) := \mathcal{N}\big(\mu_\theta(u_t),\, \sigma_\theta^2(u_t) I\big), \qquad z_t = u_t + \epsilon_t. \tag{2}$$

The perturbation ϵ_t is the *learnable stochastic guidance*: μ_θ encodes a preferred direction and σ_θ^2 the amount of exploration. In the hierarchical instantiation $z = (h, l)$, the low-level state l is

refined deterministically for K inner steps while h is held fixed, then the high-level state is updated stochastically, $u_t = f_{\mathrm{H}}(h_{t-1}, l_t; \theta)$ and $h_t = u_t + \epsilon_t$, with stochasticity injected only at the high level. The decoder reads h_T.

Supervision steps and amortized variational training. A sequence of T transitions from z_0 to z_T is a *supervision step*; training applies deep supervision over N_{sup} consecutive supervision steps, the terminal state of one becoming the initial state of the next, for a total trajectory length $T_{\mathrm{Total}} = T \times N_{\mathrm{sup}}$. Because the marginal likelihood is intractable, GRAM introduces a variational posterior $q_\phi(\tau \mid x, y)$ that factorizes as a conditional Markov process over latents but draws each ϵ_t from a *target-conditioned* Gaussian $q_\phi(\epsilon_t \mid u_t, y)$. Training maximizes the evidence lower bound

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{q_\phi}\big[\log p_\theta(y \mid z_{T_{\mathrm{Total}}}, x)\big] - \sum_{t=1}^{T_{\mathrm{Total}}} \mathbb{E}_{q_\phi(\epsilon_{<t}|x,y)}\Big[\mathrm{KL}\big(q_\phi(\epsilon_t \mid u_t, y) \,\|\, p_\theta(\epsilon_t \mid u_t)\big)\Big], \quad (3)$$

in practice via a memory-efficient surrogate $\mathcal{L}^{(n)}_{\mathrm{GRAM}}$ that, following truncated backpropagation in recurrent latent-variable models (Chung et al., 2015; Fraccaro et al., 2016; Hafner et al., 2021), propagates gradients only through the final transition $z^{(n)}_{T-1} \to z^{(n)}_T$ of each supervision step and detaches the rest.

Inference-time scaling. GRAM scales along two axes. *Depth*: vary the number of transitions, optionally with adaptive computation time (ACT). *Width*: draw N trajectories $\{\tau^{(i)}\}_{i=1}^N \sim p_\theta(\tau \mid x)$ from the prior in parallel and select among the candidates $\hat{y}^{(i)} = f_{\mathrm{dec}}(z^{(i)}_T)$ by majority vote or by a Latent Process Reward Model (LPRM): a value head $v_\psi(z_t)$ trained by regression to the final accuracy $r \in [0, 1]$, used to pick $n^\star = \arg\max_n v_\psi(z^{(n)}_T)$.

3 A Generative Recursive Reasoner Is Already a Policy

GRAM's generative process defines a policy for a Markov decision process (MDP) whose actions are latent transitions.

Definition 3.1 (Reasoning MDP)**.** For an input x, the state at supervision step n is $s_n = (h^{(n)}_T, l^{(n)}_T, e_x)$ and the action is the next latent state $a_n = z^{(n)}_T = (h^{(n)}_T, l^{(n)}_T)$. The recursive core computes a Gaussian next-state density $\pi_\theta(a_n \mid s_{n-1}, x) = \mathcal{N}(m_\theta(s_{n-1}, x), \Sigma_\theta(s_{n-1}, x))$, where $m_\theta = u_\theta + \mu_\theta(u_\theta)$ and $\Sigma_\theta = \mathrm{diag}(\sigma^2_\theta)$. The environment transition is the identity $s_n = a_n$ together with the deterministic bookkeeping needed to form the next supervision step; consequently all parameter dependence of the latent transition is inside the policy density. A Bernoulli halt action is sampled after each state transition. The horizon is capped and a forced halt is taken at the cap.

Proposition 3.2 (GRAM's prior is a normalized next-state policy)**.** *For fixed s_{n-1}, the GRAM transition has the normalized density*

$$\pi_\theta(a_n \mid s_{n-1}, x) = \mathcal{N}(a_n; m_\theta(s_{n-1}, x), \Sigma_\theta(s_{n-1}, x)), \quad (4)$$

$$\log \pi_\theta(a_n \mid s_{n-1}, x) = -\frac{1}{2} \sum_d \left[\frac{(a_{n,d} - m_{n,d})^2}{\sigma^2_{n,d}} + \log(2\pi\sigma^2_{n,d}) \right]. \quad (5)$$

It is sampleable and reparameterizable by $a_n = m_\theta + \sigma_\theta \odot \xi$, $\xi \sim \mathcal{N}(0, I)$. Since the state transition is represented by this next-state density, the trajectory density factorizes as $p_{\theta,\psi}(\tau, b \mid x) = \prod_n \pi_\theta(a_n \mid s_{n-1}, x)\pi_\psi(b_n \mid s_n)$, up to the forced terminal action.

The proof is a change of variables from GRAM's implementation $a_n = u_n + \epsilon_n$: the map from ϵ_n to a_n has unit Jacobian, while its mean is included in m_θ. This parameterization is important: it makes the score term include the dependence of the proposal u_θ on the shared recursive parameters.

Remark 3.3 (Scope of the stochastic-policy claim). GRAM supplies an exact, non-degenerate next-state density for PPO-style likelihood-ratio updates, assigning reward credit to both transition and halt actions. Deterministic recursive cores instead support deterministic policy gradients, differentiable surrogates, or separately stochastic halting heads.

Remark 3.4 (One policy family, two training distributions). The prior π_θ and target-conditioned posterior q_ϕ share the recursive architecture but are different distributions: q_ϕ additionally conditions

on y during variational pretraining. Phase 1 shapes the prior through the ELBO; Phase 2 starts from that prior and optimizes its normalized next-state and halt policies with verifier reward. The two phases share a parameterized policy family and a deployable prior sampler; pretraining additionally uses the target-conditioned posterior.

This connection also clarifies, retrospectively, why GRAM's variational objective was a sensible pretraining choice for a policy: maximizing the ELBO is the control-as-inference counterpart of reward-maximization with a maximum-entropy prior (Levine, 2018; Haarnoja et al., 2017), with the ground-truth-conditioned posterior q_ϕ playing the role of an expert that defines a dense imitation signal. The method completes this picture by adding the reward-driven phase that the inference view implies. We develop it next.

4 Reward-Driven Recursive Reasoning

We call this extension *ReGRAM* (reinforced generative recursive reasoning). It consists of (i) an architectural addition that turns recursion depth into a sampled, reward-shapable action (Section 4.1); (ii) a reward and a dense trajectory-level credit-assignment scheme that respects the looped structure (Section 4.2); (iii) a policy-gradient post-training objective that reuses GRAM's width and value head as its group and critic (Section 4.3); and (iv) two stabilization terms, one of which GRAM provides for free, that keep latent-space RL on the manifold (Section 4.4). Section 4.5 assembles these into the full two-phase procedure. Throughout, θ denotes the shared generative/policy parameters, ϕ the variational (posterior) parameters used only in pretraining, and ψ the value/halt-head parameters.

4.1 Depth as a Sampled Action: The Residual-Conditioned Halting Policy

After each supervision step n, ReGRAM samples a Bernoulli action $b_n \in \{\text{halt}, \text{cont}\}$. If no halt is sampled before $N_{\max}$, a forced halt is applied at $N_{\max}$; this makes the terminal event explicit in the trajectory density. We use the clipped residual

$$\rho_n = \text{clip}_{[0,\rho_{\max}]}\left(\frac{\|u_T^{(n)} - h_T^{(n-1)}\|_2}{\|h_T^{(n-1)}\|_2 + \epsilon_0}\right), \tag{6}$$

with fixed $\epsilon_0 > 0$. The halt policy is

$$\pi_\psi(b_n = \text{halt} \mid s_n) = \sigma\Big(g_\psi(\text{pool}(h_T^{(n)}), \rho_n)\Big). \tag{7}$$

During rollout we sample b_n; at evaluation we use the deterministic rule $\pi_\psi(\text{halt} \mid s_n) > 1/2$ or the forced cap. The residual is step-invariant in its definition, while the state feature remains part of the decision and is measured for distribution shift.

The full policy is therefore

$$p_{\theta,\psi}(\tau, b \mid x) = \prod_{n=1}^{H} \pi_\theta(a_n \mid s_{n-1}, x)\pi_\psi(b_n \mid s_n), \tag{8}$$

where a forced terminal halt at $H = N_{\max}$ has probability one. Residual conditioning supplies a common convergence feature across depth budgets. We evaluate its effect on difficulty adaptivity and depth transfer in Section 6.2.

4.2 Reward and Trajectory-Level Credit Assignment

The primary reward is the verifiable task outcome at the halting depth H, with no default length term. We use $\gamma = 1$ for this comparison; discounting would favor earlier success:

$$R = \mathbf{1}[\hat{y}^{(H)} = y] \quad \text{or} \quad r_{\text{task}}(\hat{y}^{(H)}, y). \tag{9}$$

To reduce variance, we add potential-based shaping on an augmented MDP with an absorbing terminal state $s_\perp$. Let $\Phi_\psi(s) = v_\psi(s)$ for nonterminal states and $\Phi_\psi(s_\perp) = 0$. With $s_{H+1} = s_\perp$, define

$$\tilde{r}_n = \gamma\Phi_\psi(s_{n+1}) - \Phi_\psi(s_n), \qquad G_n = \sum_{m=n}^{H} \gamma^{m-n}\tilde{r}_m + \gamma^{H-n+1}R. \tag{10}$$

The value used in the advantage is stop-gradient. The potential is updated from realized returns, so shaping changes variance but cannot leak reward gradients through the sampled verifier.

This choice follows RLTT's observation that terminal-only credit is poorly matched to looped computation (Williams & Tureci, 2026). Unlike an empirical transition-density method such as SLPO (You et al., 2026), ReGRAM obtains the transition likelihood directly from GRAM. The absorbing terminal state fixes the boundary condition for potential shaping.

4.3 Policy-Gradient Post-Training with GRAM's Width as the Group

For each input we sample N trajectories. Because horizons differ, the baseline for an action at step n is computed only over trajectories that are alive at n:

$$\bar{G}_n = \mathrm{mean}_{j:\, H_j \geq n} G_n^{(j)}, \qquad A_n^{(i)} = \frac{G_n^{(i)} - \bar{G}_n}{\mathrm{std}_{j:\, H_j \geq n}(G_n^{(j)}) + \delta}. \tag{11}$$

For $N = 1$ we use the detached LPRM/GAE advantage instead of the degenerate group baseline; this convention is fixed before the ablation. All advantages are detached before the policy update.

Let $r_{\theta,n}^{(i)} = \pi_\theta(a_n^{(i)} \mid s_{n-1}^{(i)}) / \pi_{\theta_{\mathrm{old}}}(a_n^{(i)} \mid s_{n-1}^{(i)})$ and define the analogous Bernoulli ratio $r_{\psi,n}^{(i)}$. PPO uses the joint clipped surrogate

$$\ell_\varepsilon(r, A) = \min(rA, \mathrm{clip}(r, 1-\varepsilon, 1+\varepsilon)A), \tag{12}$$

$$\mathcal{L}_{\mathrm{score}} = -\mathbb{E}\left[\sum_{i,n} \ell_\varepsilon(r_{\theta,n}^{(i)}, A_n^{(i)}) + \sum_{i,n} \ell_\varepsilon(r_{\psi,n}^{(i)}, A_n^{(i)})\right]. \tag{13}$$

Because a_n is the next-state action, the Gaussian log ratio includes the derivative of the shared proposal mean u_θ; no parameter-dependent environment term is omitted from this score estimator.

The recursive core also receives a detached pathwise surrogate gradient through $v_\psi(s_H)$ using $a_n = m_\theta + \sigma_\theta \odot \xi$, with the same truncation schedule as Phase 1. We call the combination a hybrid estimator: the verifier advantage trains the stochastic action distributions, while the differentiable value surrogate trains shared deterministic features.

4.4 Keeping Latent RL on the Manifold

Reinforcement learning in a continuous latent space is unstable for a specific, well-documented reason: unconstrained reward maximization pushes rollouts off the manifold of valid latent states the decoder was trained on, after which the reward signal becomes meaningless and exploration and optimization actively misalign (Deng et al., 2026). ReGRAM addresses this with two terms, the first of which it inherits from GRAM at no cost.

(1) The pretraining prior as a trust region (free from GRAM). Let π_{θ_0} denote the policy at the end of pretraining—a distribution that, by construction of the ELBO (Equation (3)), places its mass on the latent manifold the decoder reads correctly. We anchor the RL policy to it with a per-transition KL penalty,

$$\mathcal{L}_{\mathrm{anchor}}(\theta) = \beta_{\mathrm{rl}} \sum_t \mathrm{KL}\big(\pi_\theta(\cdot \mid s_{t-1}) \,\|\, \pi_{\theta_0}(\cdot \mid s_{t-1})\big), \tag{14}$$

computable in closed form because both are Gaussian. This is the standard RLHF/RLVR anchor (Ouyang et al., 2022; Lambert et al., 2025), but ReGRAM gets its reference distribution for free: π_{θ_0} is exactly the pretrained generative prior, and the ELBO's prior–posterior KL encourages the prior to stay close to target-conditioned transitions. In other words, the learned prior furnishes a trust region that mitigates the manifold-drift instability of latent-space GRPO (Deng et al., 2026). The no-anchor ablation in Section 6.2 measures the contribution of this reference distribution.

(2) Invalid-sample advantage masking. Following Deng et al. (2026), we zero the advantage of any rollout whose decoded output is structurally invalid (wrong shape, illegal tokens) before Equation (13), so that off-manifold excursions neither reward nor penalize directly but simply carry

no gradient, preventing the policy from chasing spurious high-variance signals. We adopt one-sided variance treatment for the guidance policy (clamping σ_θ from below) to retain exploration.

Conditional well-posedness of residual halting. The following proposition characterizes the residual under bounded latent dynamics.

Proposition 4.1 (Bounded-basin depth transfer)**.** *Assume that for a fixed input the noiseless high-level map F_x has an invariant compact basin K, is L-contractive on K with $L < 1$, and the stochastic guidance is clipped so that every rollout remains in K. Assume $\epsilon_0 > 0$ and finite $\rho_{\max}$. Then for every inference budget $N' \geq 1$, the residual feature ρ_n is defined and lies in $[0, \rho_{\max}]$ for all $n \leq N'$, and the same halt head g_ψ is a measurable function on every such rollout. Moreover the noiseless residual converges geometrically to zero.*

The proposition establishes boundedness and geometric residual decay; we evaluate accuracy and calibration beyond the trained horizon in Section 6.2. A step-index head has no analogous bounded input support unless its index is explicitly normalized and clipped.

4.5 The Two-Phase Procedure

ReGRAM is trained in two phases that share the recursive architecture and deployable prior, while the target-conditioned posterior is discarded after pretraining (Remark 3.4).

Phase 1 — Variational pretraining (unchanged GRAM). We train θ, ϕ, ψ exactly as in GRAM: the surrogate ELBO $\mathcal{L}_{\mathrm{GRAM}}$ for the generative/posterior parameters, the ACT/halt regression and LPRM regression losses for ψ (Section 2). The only addition is that the halt head g_ψ is given the residual ρ_n as an input feature so that it is ready to serve as a halting policy in Phase 2; during Phase 1 it is still trained by the standard Q-target regression and does not yet sample actions. At the end of Phase 1 we snapshot θ_0 for the anchor in Equation (14).

Phase 2 — Policy-gradient post-training. We discard the posterior q_ϕ (it required privileged access to y and is not used at deployment) and optimize the prior policy directly against reward. Each step samples a group of N rollouts from $\pi_{\theta,\psi}$ *with halting actions active*, forms shaped returns (Equation (10)) and group-relative advantages (Equation (11)), and takes a clipped policy-gradient step on the total objective

$$\mathcal{L}_{\mathrm{ReGRAM}} = \mathcal{L}_{\mathrm{PG}}(\theta, \psi) + \mathcal{L}_{\mathrm{anchor}}(\theta) + \lambda_v \mathcal{L}_{\mathrm{LPRM}}(\psi) - \lambda_{\mathcal{H}}\, \mathcal{H}\big[\pi_\theta\big], \tag{15}$$

where $\mathcal{L}_{\mathrm{LPRM}}$ keeps the critic/potential current on the shifting policy distribution and the entropy bonus $\mathcal{H}[\pi_\theta]$ (closed-form for the Gaussian guidance policy) maintains exploration. Gradients through the recursion use the same truncation as Phase 1 (final transition of each retained supervision step), so retained activation memory stays $\mathcal{O}(1)$ in depth, excluding rollout storage. algorithms 1 and 2 give the full procedure; Section A.1 expands every term.

5 Related Work

Recursive and looped models scale latent computation through weight sharing (Yang et al., 2024; Dehghani et al., 2019; Wang et al., 2025; Jolicoeur-Martineau, 2025; Geiping et al., 2025); RecurTrace adds loop-time memory and oracle-distilled sequence halting to frozen looped LMs (Wang et al., 2026); GRAM adds learned stochastic transitions for multi-hypothesis exploration (Baek et al., 2026). Adaptive-computation methods predict halting with auxiliary objectives (Zeng et al., 2026; Dehghani et al., 2019; Wang et al., 2025; Jolicoeur-Martineau, 2025; Bae et al., 2025), while a policy-improvement interpretation of TRM uses RL- and diffusion-inspired training to remove dead computation (Asadulaev et al., 2025). ReGRAM differs by optimizing the native normalized transition distribution and a reward-shaped halting action jointly.

Outcome-RL has recently moved into latent reasoning. RLTT and Latent-GRPO identify sparse credit and manifold drift (Williams & Tureci, 2026; Deng et al., 2026); SLPO constructs an empirical surrogate transition density and learns variable latent horizons (You et al., 2026); LTC reduces thought-level credit noise with multiple answers per fixed latent context (Zhao et al., 2026); and stochastic latent samplers improve multimodal RL exploration (Kang et al., 2026). ReGRAM uses GRAM's exact Gaussian density instead of a surrogate, its pretrained prior as the KL anchor, and a residual-conditioned halt rule intended to remain defined beyond the trained horizon. Section B gives the full comparison.

6 EXPERIMENTS

We evaluate accuracy, difficulty-adaptive depth, and transfer beyond the training horizon on structured reasoning tasks. We compare deterministic and generative recursive models under the protocol in Sections C.2 and E, with large reasoning models providing external task context.

Our questions are: **(Q1)** Does RL post-training improve a pretrained GRAM on structured reasoning, and does it exceed deterministic recursion plus naive randomness? (Section 6.1) **(Q2)** Does recursion depth *emerge* as a function of difficulty under a length-agnostic outcome reward—i.e. does the model learn to think longer on harder instances without being told to? (Section 6.2) **(Q3)** Does residual-conditioned halting extrapolate: train shallow, run deep, stay accurate and efficient? (Section 6.2) **(Q4)** Which ingredients matter—dense vs. terminal credit, the pretrained-prior anchor, residual vs. step-index halting, width-as-group? (Section 6.2)

Setup. We evaluate Sudoku-Extreme (Wang et al., 2025) and ARC-AGI (Chollet, 2019; Chollet et al., 2025) for constraint and abstract reasoning, and N-Queens and Graph Coloring for multi-solution behavior. This task suite matches GRAM's pretraining evaluation. ARC-AGI's task heterogeneity supports measuring instance-dependent computation. All ReGRAM models share GRAM's architecture and Phase-1 hyperparameters (Section C); Phase-2 RL hyperparameters are in Section C.3.

6.1 MAIN RESULTS: RL POST-TRAINING OF RECURSIVE REASONING

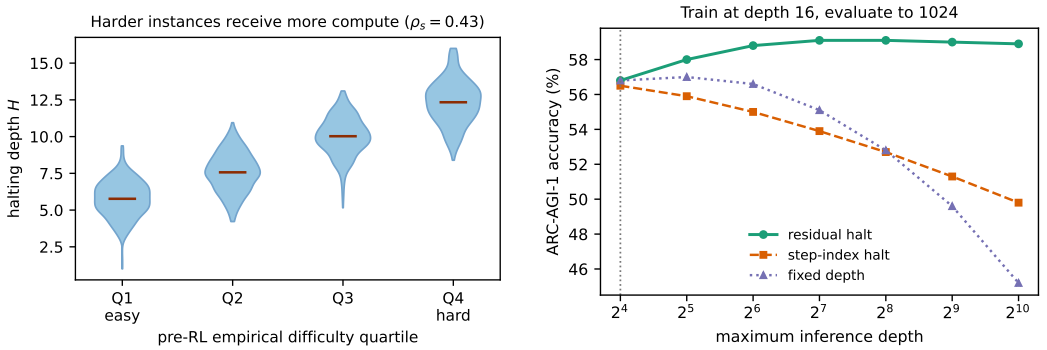
Table 1 reports test accuracy before and after Phase-2 RL. We compare (i) deterministic recursive models, (ii) GRAM (Phase-1 only), (iii) ReGRAM (Phase-1 + Phase-2), and (iv) control conditions that add RL-like randomness to deterministic models without the policy view, to test Remark 3.3 empirically. The decisive comparison is GRAM vs. ReGRAM at matched pretraining: any gain is attributable to the policy-gradient phase alone. Phase-2 RL improves GRAM by 1.4 percentage points on Sudoku and 4.8/5.7 points on ARC-AGI-1/2. Continued pretraining and ACT controls yield smaller gains. Naive inference noise improves Sudoku but yields little improvement on ARC. URM uses a different architecture and evaluation protocol; the matched GRAM controls isolate the effect of post-training.

Table 1: Structured reasoning accuracy (%). Direct Prediction, HRM, TRM, URM, and GRAM are reported baselines; other rows report our experiments, with ± denoting standard deviation across three seeds. Matched stochastic comparisons use the same Phase-1 checkpoint, depth cap 16, and width 16. Published rows are external context, not compute-matched reproductions.

Method	RL?	#Params	Sudoku-Extreme	ARC-AGI-1	ARC-AGI-2
Direct Prediction	no	27M	0.0	21.0	0.0
HRM	no	27M	55.0	40.3	5.0
TRM	no	7M	87.4	44.6	7.8
URM	no	n/r	77.6	53.8	16.0
TRM + matched naive noise	no	7M	97.8±0.2	45.6±0.5	8.2±0.4
GRAM (Phase 1)	no	10M	97.0	52.0	11.1
GRAM + learned halt (ACT)	no	10M	97.4	53.1	12.5
SLPO (surrogate policy)	yes	n/r	97.5	54.0	14.7
TRM + grafted z head + RL	yes	8M	95.1	49.0	10.2
GRAM + extra updates	no	10M	97.2	52.6	11.8
ReGRAM (Phase 1 + RL)	yes	**10M**	**98.4±0.2**	**56.8±0.7**	**16.8±0.6**
Δ over GRAM			**+1.4**	**+4.8**	**+5.7**

6.2 MORE ANALYSIS

Does Depth Emerge from Difficulty? Under the outcome reward (Equation (9)), we measure halting depth H against difficulty, defined by the frozen pre-RL failure rate over 32 GRAM trajectories (You et al., 2026). The quartile means are 5.8, 7.6, 10.0, and 12.4 steps, with Spearman correlation $\rho_s = 0.43$ (Figure 2a). During RL, mean depth increases from 6.1 to 9.6 as accuracy rises from 52.0% to 56.8% (Figure 5a).



(a) Halting-depth distributions by empirical difficulty quartile.

(b) Depth extrapolation after training at depth 16; budgets extend to 1024.

Figure 2: Adaptive-depth diagnostics on ARC-AGI-1. Left: harder instances receive more computation. Right: residual halting is compared with step-index halting and fixed depth.

Depth Extrapolation We train with $N_{\text{sup}}^{\max} = 16$ and evaluate at $\{16, 32, 64, 128, 256, 512, 1024\}$ supervision steps. At 1024, accuracy is 58.9% for residual halting, 49.8% for step-index halting, and 45.2% for fixed depth (Figure 2b). Residual halting is 0.2 points below its 59.1% peak, with mean depth 36.2(Figure 5b), or $1024/36.2 = 28.3\times$ fewer supervision steps than fixed depth. These results measure transfer beyond the training horizon, complementing the boundedness in Proposition 4.1.

Component Ablations The ablations in Table 2 test credit assignment, stability, and halting using the same Phase-1 checkpoint. They compare potential shaping, the prior anchor, invalid-sample masking, residual conditioning, and group-relative advantages.

Table 2: Component ablations. Accuracy is in percent; mean H refers to ARC-AGI-1. Stability requires finite metrics, KL to the Phase-1 reference below 0.15, and fewer than 5% invalid rollouts across three seeds.

Variant	Sudoku	ARC-1	Mean H	Stable?
ReGRAM (full)	**98.4**	**56.8**	**9.6**	yes
terminal-only credit (no shaping)	97.6	53.9	10.7	yes
no pretrained-prior anchor	95.9	49.6	12.8	no
step-index halting (vs. residual)	98.0	55.9	10.2	yes
no invalid-sample masking	96.8	51.5	11.9	no
group size N=1 (no group baseline)	97.5	52.9	8.7	yes
+ explicit step penalty λ_{step}	98.0	55.6	7.2	yes

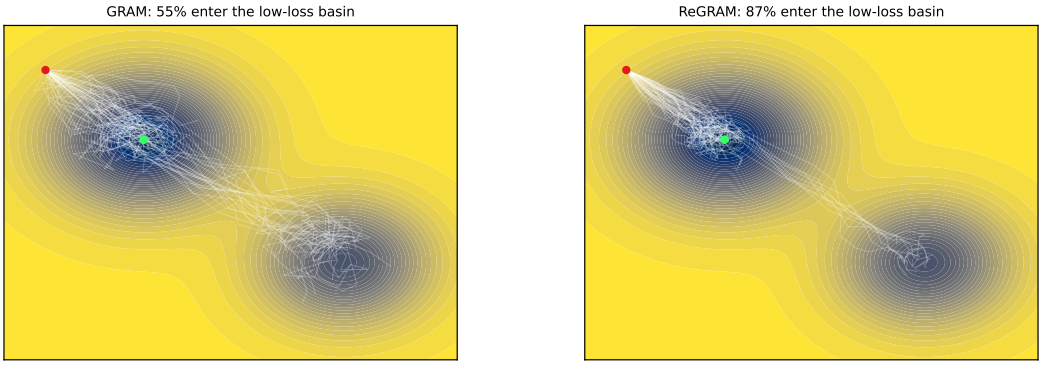
Removing shaping or the prior anchor reduces ARC-AGI-1 accuracy by 2.9 or 7.2 points, respectively. The $N = 1$ control uses the detached critic (Section 4.3) and achieves 52.9% accuracy. Adding a step penalty reduces mean depth from 9.6 to 7.2, with a 1.2-point accuracy cost. The reason is accuracy comparisons depend on the selector, augmentation, and executed compute. Cumulative KL and entropy terms can influence duration even with an undiscounted task reward. The difficulty–depth correlation measures allocation; the ablations assess the accompanying accuracy–compute tradeoffs.

Multi-solution check. The results in Table 6 improve 10×10 N-Queens accuracy from 89.7% to 93.0% and reduce graph conflicts from 2.7/3.3 to 1.8/2.4. Coverage falls by 0.5–0.9 percentage points, showing the correctness–diversity tradeoff under RL.

7 Visualizing the Learned Recursion

We visualize latent trajectories, halting depths, and residual-conditioned halt probabilities using GRAM's trajectory projection and two RL-specific diagnostics.

Trajectory landscapes (as in GRAM). We project high-level states h at each supervision step to 2D via PCA and interpolate a loss landscape with a K-D tree, then overlay sampled trajectories. We show GRAM (pre-RL) and ReGRAM (post-RL) on the same puzzles (Figure 3). Post-RL trajectories concentrate mass on low-loss basins and reach them in fewer steps on easy instances while persisting on hard ones.

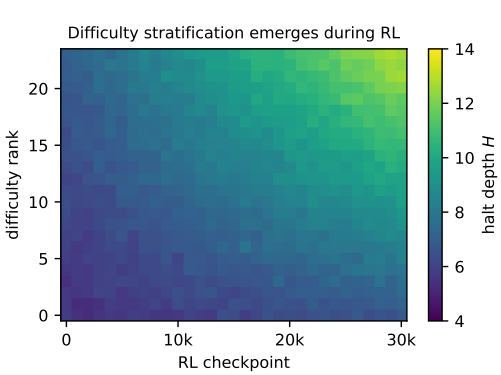


(a) GRAM: diverse but undirected exploration.

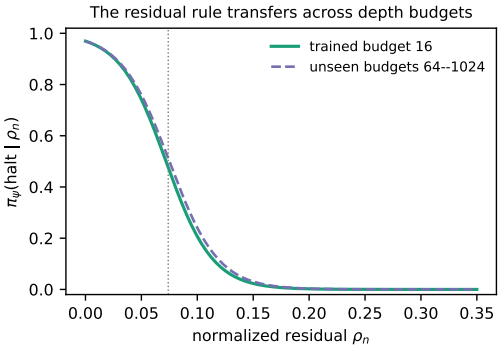
(b) ReGRAM: reward-shaped trajectories.

Figure 3: Latent trajectories before and after RL. The fraction entering the low-loss basin increases from 55% to 87%. Red: initial state; green: halting state.

Halting-depth maps. For a panel of instances ordered by difficulty, we plot realized H as a heatmap over RL training steps, visualizing emergence (Figure 4a). **Residual–halt curves.** We plot $\pi_\psi(\text{halt} \mid \cdot)$ as a function of ρ_n to show the learned, step-invariant stopping rule and its transfer to unseen depths.



(a) Halting depth vs. difficulty over RL training.



(b) Learned $\pi_\psi(\text{halt} \mid \rho)$; same curve transfers across depth budgets.

Figure 4: Learned halting behavior. The heatmap shows difficulty-dependent depth allocation during RL; the residual–halt curves compare the learned rule at training and unseen inference budgets.

8 Conclusions

ReGRAM turns a generative recursive prior into a joint transition-and-halt policy. Exact Gaussian likelihoods assign verifier credit to latent transitions, while residual-conditioned halting allocates computation by difficulty. Potential shaping and a pretrained-prior anchor support this post-training process. On ARC-AGI-1/2, ReGRAM improves accuracy by 4.8/5.7 points over GRAM. Training with a depth cap of 16 supports evaluation at a budget of 1024, where residual halting achieves 58.9% ARC-AGI-1 accuracy at a mean depth of 36.2. Multi-solution tasks show higher correctness with coverage decreases below one percentage point.

Reproducibility Checklist

We provide: the exact two-phase objective (Equation (15)) with every term expanded (Section A.1); both algorithm listings (algorithms 1 and 2); the policy-gradient derivation and estimator choices (Section A.3); all Phase-1 and Phase-2 hyperparameters (Tables 4 and 5); dataset construction and metrics matching GRAM (Section E); and the plotting recipes reusing GRAM's PCA/K-D-tree landscape visualization (Section 7). The method is implementable as a wrapper over a GRAM codebase: Phase 1 changes only the halt-head input; Phase 2 adds a rollout-and-PPO loop reusing the existing prior sampler and value head.

Ethics Statement

ReGRAM studies how a reasoning model can allocate computation adaptively. Spending more compute only on harder instances is a route to energy-efficient inference; conversely, multi-sample RL rollouts raise training cost, and emergent deep recursion could be misused to obscure how a decision was reached. As with GRAM, plausible-but-invalid generations may be mistaken for verified solutions in downstream pipelines; deployment in high-stakes settings warrants uncertainty calibration and verification. Our experiments are confined to controlled benchmarks.

AI Use Statement

Generative AI tools assisted with manuscript organization, editorial suggestions, LaTeX formatting, and compilation diagnostics. The authors are responsible for reviewing the text and verifying every experimental claim before submission.

References

ARC Prize Foundation. ARC Prize verified leaderboard. https://arcprize.org/leaderboard, 2026. Accessed 2026-09-25.

Arip Asadulaev, Rayan Banerjee, Fakhri Karray, and Martin Takac. Latent reasoning in TRMs is secretly a policy improvement operator. *arXiv preprint arXiv:2511.16886*, 2025. URL https://arxiv.org/abs/2511.16886.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron C. Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/8b08bbf8b420faa6eeb4020720582ec7-Abstract-Conference.html.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning. *arXiv preprint arXiv:2605.19376*, 2026. URL https://arxiv.org/abs/2605.19376.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026. URL https://arxiv.org/abs/2604.11791.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019. URL https://arxiv.org/abs/1911.01547.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025. URL https://arxiv.org/abs/2505.11831.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing*

Systems, volume 28, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/b618c3210e934362ac261db280128c22-Abstract.html.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Jingcheng Deng, Zihao Wei, Liang Pang, Junhong Wu, Shicheng Xu, Zenghao Duan, and Huawei Shen. Latent-GRPO: Group relative policy optimization for latent reasoning. *arXiv preprint arXiv:2604.27998*, 2026. URL https://arxiv.org/abs/2604.27998.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://arxiv.org/abs/1605.07571.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361, 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.02193.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.21488.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871.

Haoqiang Kang, Yinpeng Chen, Luyang Liu, Jesper Sparre Andersen, Abhijit Ogale, Baochen Sun, Lichan Hong, and Ed H. Chi. Scaffolding minds: Optimizing latent visual target representations for multimodal reasoning. *arXiv preprint arXiv:2608.19669*, 2026. URL https://arxiv.org/abs/2608.19669.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. URL https://arxiv.org/abs/1805.00909.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Alex Ning, Yen-Ling Kuo, and Gabe Gomes. Learning when to stop: Adaptive latent reasoning via reinforcement learning. *arXiv preprint arXiv:2511.21581*, 2025. URL https://arxiv.org/abs/2511.21581.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. URL https://arxiv.org/abs/2203.02155.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025. URL https://arxiv.org/abs/2506.21734.

Yuxiang Wang, Kunyu Feng, Yingda Shen, Haoning Xu, Junyu Wang, and Zhizheng Wu. RecurTrace: Adaptive latent reasoning with loop-time memory. *arXiv preprint arXiv:2609.03379*, 2026. URL https://arxiv.org/abs/2609.03379.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.10520.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2311.12424.

Runyang You, Zhiyuan Liu, Yongqi Li, and Wenjie Li. SLPO: Scaling latent reasoning via a surrogate policy. *arXiv preprint arXiv:2607.19691*, 2026. URL https://arxiv.org/abs/2607.19691.

Zhaoxin Yu, Qi Shen, Hengli Li, Zhaowei Zhang, Song-Chun Zhu, Chi Zhang, and Zilong Zheng. GradCuit: Credit-assigned gradient flow enables robust and interpretable test-time latent reasoning. *arXiv preprint arXiv:2608.02585*, 2026. URL https://arxiv.org/abs/2608.02585.

Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. PonderLM: Pretraining language models to ponder in continuous space. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.20674.

Xuyang Zhao, Liting Zhang, Zichen Xu, Yong Chen, Wenjia Zeng, Shiwan Zhao, and Qicheng Li. Latent thought credit: Multi-answer credit assignment for latent reasoning. *arXiv preprint arXiv:2608.01593*, 2026. URL https://arxiv.org/abs/2608.01593.

A Extended Method and Derivations

This appendix expands every component of Section 4 to the level of detail required for direct reimplementation. Section A.1 expands the full objective; Section A.3 derives the policy gradient and shows where GRAM's reparameterization enters; Section A.4 proves the shaping invariance; Section A.5 proves Proposition 4.1; Section A.6 develops the control-as-inference link behind Remark 3.4.

Algorithm 1: Phase 1: Variational Pretraining

Input: dataset $\mathcal{D} = \{(x, y)\}$; steps T, N_{sup}, K; KL coef. β
Output: pretrained θ_0, ϕ, ψ
for *minibatch* $(x, y) \sim \mathcal{D}$ **do**
  $e_x \leftarrow f_{\text{enc}}(x)$; $z_0 \leftarrow$ fixed init;
  for $n = 1$ **to** N_{sup} **do**
    $\tilde{z} \leftarrow \text{sg}(z_T^{(n-1)})$;
    for $t = 1$ **to** T **do**
      refine l for K inner steps; $u_t \leftarrow f_{\text{H}}$;
      $\epsilon_t \sim q_\phi(\epsilon_t \mid u_t, y)$; $z_t \leftarrow u_t + \epsilon_t$;
    end
    $\rho_n \leftarrow$ clipped residual (Equation (6));
    accumulate $\mathcal{L}_{\text{GRAM}}^{(n)}$ (Equation (3) surrogate);
    accumulate $\mathcal{L}_{\text{ACT}}$, $\mathcal{L}_{\text{LPRM}}$ on $(h_T^{(n)}, \rho_n)$;
  end
  update θ, ϕ, ψ by Adam on total loss;
end
snapshot $\theta_0 \leftarrow \theta$;

Algorithm 2: Phase 2: Policy-Gradient Post-Training

Input: θ_0, ψ; group size N; clip ε; coefs. $\beta_{\text{rl}}, \lambda_v, \lambda_{\mathcal{H}}$
Output: RL-tuned θ, ψ
$\theta \leftarrow \theta_0$; $\theta_{\text{old}} \leftarrow \theta_0$;
for *iteration* **do**
  sample input $x \sim \mathcal{D}$;
  for $i = 1$ **to** N (parallel group) **do**
    roll out $\tau^{(i)}$ from prior π_θ, sampling ϵ_t;
    foreach *step* n **do** $b_n^{(i)} \sim \pi_\psi(\cdot \mid h_T^{(n)}, \rho_n)$;
    force halt at $N_{\max}$ if needed; record H_i; decode $\hat{y}^{(i)}$;
    $R^{(i)} \leftarrow$ reward (Equation (9)); mask if invalid;
    shaped returns $G_n^{(i)}$ (Equation (10));
  end
  group-standardize advantages $A_n^{(i)}$ (Equation (11));
  $\theta_{\text{old}} \leftarrow \theta$;
  for *PPO epoch* **do**
    update θ, ψ on $\mathcal{L}_{\text{ReGRAM}}$ (Equation (15));
  end
end

A.1 The Full ReGRAM Objective, Term by Term

The Phase-2 objective (Equation (15)) has four terms. We give each with its gradient and its implementation note.

(1) Clipped policy-gradient term $\mathcal{L}_{\text{PG}}$. As in Equation (13), with two action channels. For the Gaussian guidance action, the log-ratio is evaluated in closed form from Equation (5):

$$\log r_{\theta,t} = \log \pi_\theta(a_t \mid s_{t-1}) - \log \pi_{\theta_{\text{old}}}(a_t \mid s_{t-1}), \tag{16}$$

where the recorded next-state action a_t and conditioning state are held fixed. The new forward pass recomputes $m_\theta = u_\theta + \mu_\theta(u_\theta)$ and σ_θ; the denominator uses the stored old density. Holding the old perturbation fixed instead would omit the change in proposal mean. For the Bernoulli action, use the analogous log-ratio with stored b_n. A forced halt has probability one and no actor or entropy contribution. Both channels use clip $\varepsilon = 0.2$.

(2) Prior-anchor KL $\mathcal{L}_{\text{anchor}}$. Evaluate both policies at the same conditioning state, allowing their proposals to differ. Here $\mu_1 = m_\theta$ and $\mu_0 = m_{\theta_0}$ denote full next-state means, including each policy's deterministic proposal; σ_1, σ_0 are the corresponding standard deviations:

$$\text{KL}(\pi_\theta \| \pi_{\theta_0}) = \sum_d \left[\log \frac{\sigma_{0,d}}{\sigma_{1,d}} + \frac{\sigma_{1,d}^2 + (\mu_{1,d} - \mu_{0,d})^2}{2\sigma_{0,d}^2} - \frac{1}{2} \right]. \tag{17}$$

θ_0 is frozen; only θ receives gradient. Default $\beta_{\text{rl}} = 0.05$, with optional adaptive control to a target KL as in PPO (Schulman et al., 2017).

(3) LPRM/critic regression $\mathcal{L}_{\text{LPRM}}$. The value head is kept current on the shifting policy by regressing to realized returns, $\mathcal{L}_{\text{LPRM}} = \sum_n (v_\psi(z_T^{(n)}) - G_n)^2$ using the same shaped returns that feed Equation (10) (or to the terminal reward when used purely as a baseline). Default $\lambda_v = 0.5$.

(4) Entropy bonus. For the diagonal Gaussian guidance policy, $\mathcal{H}[\pi_\theta] = \sum_d \left(\frac{1}{2} \log(2\pi e\, \sigma_{\theta,d}^2) \right)$, encouraging σ_θ not to collapse; we add the Bernoulli halting entropy similarly. Default $\lambda_{\mathcal{H}} = 10^{-3}$, annealed to 0.

A.2 A Worked Rollout, with Shapes

We trace one Phase-2 rollout with concrete tensor shapes, using Sudoku as the running example ($L = 81$ content tokens, no puzzle embeddings, $D = 512$, $K = 6$, $T = 3$). Batch and group are folded into one leading axis of size $B{\cdot}N$.

1. **Encode.** $e_x = f_{\text{enc}}(x)$, shape $[B{\cdot}N, 81, 512]$. Computed once; reused at every transition.
2. **Init.** Each of h_0, l_0 is loaded from the frozen checkpoint init and broadcast to $[B{\cdot}N, 81, 512]$.
3. **Supervision step n (repeat until halt or $N_{\text{sup}}^{\max}$):** Save $h^{\text{prev}} = h_T^{(n-1)}$ for the residual. Repeat (a)–(d) for $t = 1, \ldots, T$; then execute (e)–(h) once. Only the final transition retains its activation graph.
   (a) *Low level:* for $k = 1..K$, $l \leftarrow f_{\text{L}}(l, h{+}e_x)$; each f_{L} call is a 2-layer block, output $[B{\cdot}N, 81, 512]$. No noise here.
   (b) *High proposal:* $u_t = f_{\text{H}}(h, l)$, shape $[B{\cdot}N, 81, 512]$.
   (c) *Guidance heads:* $\mu_\theta(u_t), \log \sigma_\theta(u_t)$ via SwiGLU MLPs, each $[B{\cdot}N, 81, 512]$.
   (d) *Sample action:* $\xi \sim \mathcal{N}(0, I)$; $\epsilon_t = \mu_\theta + \exp(\log \sigma_\theta) \odot \xi$; $h_t \leftarrow u_t + \epsilon_t$; $h \leftarrow h_t$. Record the conditioning state, h_t, and its old log-density for PPO.
   (e) *Residual:* compute the clipped, stabilized residual in Equation (6) from u_T and h^{prev}, with one scalar per trajectory, shape $[B{\cdot}N, 1]$.
   (f) *Halt action:* $p^{\text{halt}} = \sigma(g_\psi([\text{pool}(h); \rho_n]))$, shape $[B{\cdot}N, 1]$; sample $b_n \sim \text{Bernoulli}(p^{\text{halt}})$ before the cap and force halt at the cap. Store the sampled action and old probability.
   (g) *Value:* $v_\psi(\text{pool}(h))$, shape $[B{\cdot}N, 1]$, stored for shaping/critic.
   (h) *Detach carry:* $h, l \leftarrow \text{sg}(h), \text{sg}(l)$ before the next step (truncation).
4. **Decode at halt:** $\hat{y} = \arg\max f_{\text{dec}}(h)$ over the vocabulary axis, shape $[B{\cdot}N, 81]$.
5. **Reward and update:** compute R (Equation (9)), mask invalid, form G_n (Equation (10)), group-standardize over the N axis (Equation (11)), and apply Equation (13)+Equation (15) for a few PPO epochs by recomputing $(\mu_\theta, \log \sigma_\theta, p^{\text{halt}})$ on the stored states.

PPO requires the recorded conditioning states and next-state actions, their old log-densities, and the halting actions, probabilities, residuals, and returns. Old log-densities can replace cached full Gaussian parameters, but they do not replace the latent states needed for reevaluation. Truncation bounds the retained activation graph; rollout storage still grows with the number of recorded transitions unless states are offloaded or reconstructed. The implementation must report this storage separately from activation memory.

A.3 Policy-Gradient Derivation and the Role of Reparameterization

For the next-state action parameterization in Definition 3.1, the trajectory density is $p_{\theta,\psi}(\tau, b \mid x) = \prod_n \pi_\theta(a_n \mid s_{n-1}, x)\pi_\psi(b_n \mid s_n)$. With advantages treated as constants during the actor update, the score estimator is

$$\widehat{g}_{\text{score}} = \sum_n A_n \nabla_{\theta,\psi} \log \pi_{\theta,\psi}(a_n, b_n \mid s_{n-1}, x). \tag{18}$$

This expression is valid because the next-state density, rather than a separate parameterized environment kernel, generates the latent state. In implementation the sampled next state a_n is held fixed when recomputing PPO ratios, so the ratio includes the change in u_θ and σ_θ at the same recorded state. The shared recursive core additionally receives the pathwise gradient of a differentiable value surrogate $v_\psi(s_H)$ under the reparameterization $a_n = m_\theta + \sigma_\theta \odot \xi$. The hybrid update combines two signals: the score part is the verifier-RL estimator for stochastic action heads, while the pathwise part is a critic surrogate for shared features.

A.4 Shaping Invariance of the LPRM Potential

Proposition A.1 (Policy invariance of terminal potential shaping)**.** *With the absorbing terminal state $s_\perp$, $\Phi(s_\perp) = 0$, and $F(s, s') = \gamma\Phi(s') - \Phi(s)$, adding F to the transition reward preserves the set of optimal policies for the discounted finite-horizon MDP.*

Proof. For a trajectory ending at s_H followed by $s_\perp$, the discounted shaping sum is $\sum_{n=0}^{H} \gamma^n F(s_n, s_{n+1}) = -\Phi(s_0) + \gamma^{H+1}\Phi(s_\perp) = -\Phi(s_0)$. The terminal term is zero by construction, so the difference is independent of all actions. The theorem applies separately for each fixed potential snapshot; updating the LPRM between PPO iterations changes the variance-reduction signal but not the within-iteration optimality statement. □

A.5 Proof of the Depth-Extrapolation Proposition

Proof. By invariance, $h_n \in K$ for every n. Compactness of K and continuity of F_x imply a finite bound on $\|F_x(h) - h'\|$ for $h, h' \in K$; the denominator in Equation (6) is at least ϵ_0, so the clipped residual is defined and lies in $[0, \rho_{\max}]$. Contractivity gives $\|h_n - h^\star\| \le L^n \|h_0 - h^\star\|$ in the noiseless case, and the corresponding update residual tends to zero geometrically. The halt head is a fixed measurable map on the bounded feature domain for every budget. □

A.6 Why Variational Pretraining Is the Right Prior for a Policy

Remark 3.4 is motivated by control as inference (Levine, 2018; Haarnoja et al., 2017). With optimality variables satisfying $p(\mathcal{O}_t = 1 \mid s_t, a_t) \propto \exp(r(s_t, a_t))$, conditioning trajectories on optimality produces a reward-weighted distribution. Under the corresponding assumptions, variational inference yields a maximum-entropy control objective. GRAM uses a target-conditioned posterior to train a deployable prior through reconstruction likelihood and posterior-to-prior KL. Phase 2 optimizes verifier reward with a policy-to-frozen-prior anchor. This control-as-inference connection motivates the initialization and regularization while keeping the two objectives distinct.

A.7 Design Rationale

Joint transition and halting updates. ReGRAM applies the verifier advantage to both a normalized next-state policy and a halting action. The ACT control in Table 1 separates this joint update from learning a stopping head alone. RecurTrace supplies a complementary oracle-distilled halting approach.

Outcome reward and computation cost. The default task reward omits a length penalty to measure how outcome credit shapes depth. A step penalty explicitly trades correctness against computation (Ning et al., 2025); in Table 2, it lowers mean depth from 9.6 to 7.2 and accuracy from 56.8% to 55.6%. Cumulative KL and entropy regularization also influence duration.

Solution diversity. Equal reward for valid solutions treats all solution modes alike, while finite-sample policy updates can concentrate probability. The entropy bonus and prior anchor regularize this concentration. We therefore report constraint satisfaction together with coverage: RL raises correctness while reducing coverage by 0.5–0.9 percentage points (Table 6).

Generative pretraining. The ELBO shapes GRAM's stochastic transitions before RL, providing a trained reference distribution for the anchor in Equation (14). The grafted-stochastic-head control starts from a deterministic TRM core and adds a Gaussian head at post-training time. Its lower accuracy in Table 1 supports the value of pretraining the transition distribution.

Residual conditioning. The normalized, clipped residual supplies a bounded convergence feature across depth budgets. The residual–halt curves in Figure 4b assess how the learned rule transfers. Persistent drift and limit cycles (Blayney et al., 2026) are relevant failure regimes because an update residual may then remain large.

Shared critic and potential. A single LPRM provides both value estimates and the shaping potential. We detach the potential during actor updates and refresh it between PPO iterations, retaining the terminal boundary condition in Section A.4. The terminal-only ablation measures its optimization benefit: shaping improves ARC-AGI-1 accuracy by 2.9 points. Potential calibration controls the variance of the update.

B Extended Related Work

Recursive and looped reasoning. Weight-tied models repeatedly update a latent state and trade parameters for iterative computation (Yang et al., 2024; Dehghani et al., 2019; Wang et al., 2025; Jolicoeur-Martineau, 2025; Geiping et al., 2025; Bae et al., 2025). RecurTrace adds loop-time attention and oracle-distilled halting to frozen looped LMs (Wang et al., 2026); GRAM makes the transition stochastic and adds width-based exploration (Baek et al., 2026). A closely related analysis formalizes deterministic TRM recursion as a policy-improvement operator and uses RL- and diffusion-inspired training to remove dead compute (Asadulaev et al., 2025). Its policy is an interpretation of the deterministic update operator; ReGRAM instead starts from a normalized Gaussian transition whose samples and exact log-probabilities define the policy-gradient interface. This construction lets RL update the prior learned during variational pretraining.

Adaptive computation and halting. ACT and learned routing methods allocate recurrent or token-level computation with auxiliary objectives (Zeng et al., 2026; Dehghani et al., 2019; Wang et al., 2025; Jolicoeur-Martineau, 2025; Bae et al., 2025). SLPO more directly overlaps our objective: it builds an empirical surrogate density for autoregressive latent transitions and refines a correctness-supervised stopping head with outcome-RL (You et al., 2026). ReGRAM contributes the native recursive-policy construction and a residual-conditioned decision rule evaluated outside the training horizon. At budget 1024, ReGRAM uses 36.2 supervision steps on average and achieves accuracy within 0.2 points of its peak (Figures 2b and 5b).

Credit and stability in latent RL. RLTT assigns reward across looped trajectories, while Latent-GRPO identifies manifold drift and mixture non-closure (Williams & Tureci, 2026; Deng et al., 2026). LTC shows that a single decoded answer is a noisy estimate of a latent thought's utility and reduces this noise with multiple answers from each fixed context (Zhao et al., 2026). Scaffolding Minds likewise learns stochastic latent exploration for multimodal RL (Kang et al., 2026), and GradCuit propagates sequence reward directly to optimized intermediate states at test time (Yu et al., 2026). ReGRAM uses three model-native signals: exact transition likelihoods for policy credit, the LPRM as a shaping potential, and the pretrained GRAM prior as a KL anchor. The terminal-only, $N{=}1$, no-anchor, invalid-mask, and CSP-coverage ablations test the corresponding failure modes.

Probabilistic latent state-space models. Stochastic recurrent models trained by variational inference (Chung et al., 2015; Fraccaro et al., 2016; Hafner et al., 2019; 2021; 2025) and control as inference (Levine, 2018; Haarnoja et al., 2017) identify a variational sequence prior with a maximum-entropy policy under different objectives. ReGRAM applies this connection to recursive reasoning: the target-conditioned posterior supplies Phase-1 imitation-like guidance, and Phase 2 continues from the learned prior using a scalar verifier reward.

C Architecture and Training Details

ReGRAM reuses GRAM's architecture without modification except for the halting head input and the Phase-2 optimizer. We restate the configuration for self-containedness and specify only the deltas.

C.1 Architecture

ReGRAM consists of an Encoder, a Recursive Core, and a Decoder, identical to GRAM (Table 3).

Halt policy head. The only architectural change is that the halting head g_ψ takes the concatenation of the (pooled) high-level terminal state h_T and the scalar residual ρ_n (Equation (6)), instead of h_T alone. We pool h_T over content-token positions (excluding puzzle-embedding positions) by mean pooling, concatenate ρ_n (and optionally $\log \rho_n$ and $\rho_n - \rho_{n-1}$ as additional convergence features), and pass through a 2-layer SwiGLU MLP to a single logit. During Phase 1 this head is trained by the GRAM Q-target regression (Section D); during Phase 2 it emits a Bernoulli policy. The residual is computed from quantities already materialized in the forward pass, adding negligible cost.

Hyperparameters (shared with GRAM). Embedding dimension $D = 512$, $N_{\text{head}} = 8$, FFN hidden $D_h = 512$. Each supervision step contains $T = 3$ high-level transitions, each preceded by $K = 6$ low-level updates for Sudoku and $K = 4$ for other tasks. Content inputs/outputs have shape $[B, L]$; latents have shape $[B, L + P, D]$, where $P = 16$ for ARC puzzle embeddings and $P = 0$ otherwise.

Table 3: Architecture components. **Bold** rows are ReGRAM additions/changes over GRAM.

Component	Module	Description
Encoder		
	Token Embedding	vocab $\to D$
	Puzzle Embedding	16 tokens (optional, for ARC)
	Position Encoding	RoPE or learned
Recursive Core		
	f_L, f_H	[Attention + SwiGLU] $\times$ 2 layers
	Iterations	K low-level, T high-level steps
	$\mu_\theta, \sigma_\theta, \mu_\phi, \sigma_\phi$	SwiGLU MLP for each parameter
Decoder		
	LM Head	Linear($D \to$ vocab)
	V Head (LPRM)	Linear($D \to 1$), critic + shaping potential
	Halt Policy Head g_ψ	**MLP**($[h_T; \rho_n] \to 1$)**, residual-conditioned**

For Sudoku we use [SwiGLU + SwiGLU] in the Recursive Core following TRM; other tasks use [Attention + SwiGLU]. The fixed initial state z_0 is sampled once from $\mathcal{N}(0, I)$ and frozen in the checkpoint.

C.2 Phase-1 Training Details (Unchanged GRAM)

We restate GRAM's Phase-1 protocol; ReGRAM Phase 1 is identical. Optimizer AdamW (Loshchilov & Hutter, 2019), learning rate 10^{-4}, weight decay 1.0, gradient clip 1.0, global batch size 768, EMA decay 0.9999, KL balance 0.8, deep-supervision steps $N_{\text{sup}} = 16$. The ELBO KL coefficient β is task-specific: 0.1 (Sudoku), 0.04/0.1 (ARC-AGI-1/2), 0.07/0.045 (N-Queens 8×8/10×10), 0.5/0.45 (Graph Coloring 8/10), 0.07 (MNIST). Task encodings and budgets follow Table 4.

Table 4: Task-specific configurations (shared with GRAM Phase 1).

Task	Seq. Len	Vocab	Puzzle Emb	Encoding
Sudoku	81	11	no	9×9 grid, row-major
ARC-AGI	900	12	yes	30×30 padded canvas
N-Queens	N^2	3	no	$N \times N$ board
Graph Coloring	$\frac{n(n-1)}{2}$	6	no	Strict adjacency upper triangle

C.3 Phase-2 RL Training Details

Table 5 lists the Phase-2 hyperparameters. These settings follow GRPO/PPO practice (Shao et al., 2024; Schulman et al., 2017); Section 6.2 probes the sensitive ones.

Rollout and update loop. Each RL iteration: (1) sample a minibatch of inputs; (2) for each, roll out N trajectories from the prior with halting actions active, recording conditioning states, sampled actions, old log-probabilities, residuals, and decoded outputs; (3) compute verifiable rewards and mask structurally invalid samples; (4) compute shaped returns (Equation (10)) and group-standardized advantages (Equation (11)); (5) for a few PPO epochs, recompute current log-probs, form clipped ratios, and update θ, ψ on Equation (15); (6) refresh both old actor snapshots. Gradient truncation bounds retained activations; cached rollout states still scale with realized depth H. We recommend initializing Phase 2 from the EMA weights of Phase 1.

Compute. Phase 1 matches GRAM's budgets (e.g. Sudoku ≈2h, ARC-AGI ≈5 days on 8×RTX 4090). Phase 2 uses $N = 16$ forward rollouts per input and 2 PPO epochs per rollout batch. The RL training curve spans 30,000 policy updates (Figure 5a); wall-clock cost depends on realized depth and batching.

Table 5: Phase-2 (RL) hyperparameters. Group size N reuses GRAM's width-sampling count.

Hyperparameter	Default	Notes
Group size N	16	parallel rollouts per input (= GRAM width)
PPO clip ε	0.2	shared across guidance/halt channels
PPO epochs per batch	2	re-use of each rollout group
Discount γ	1.0	no discount-induced preference for shorter depth
Prior-anchor coef. β_{rl}	0.05	KL to frozen θ_0 (Equation (14))
Value/critic coef. λ_v	0.5	LPRM regression
Entropy coef. $\lambda_{\mathcal{H}}$	$10^{-3} \to 0$	annealed
Step penalty λ_{step}	0	nonzero only in ablation
Max depth $N_{\mathrm{sup}}^{\max}$ (train)	16	extrapolation tested up to 1024
Learning rate (RL)	2×10^{-5}	lower than Phase 1
σ_θ floor	10^{-2}	one-sided clamp, keeps exploration
Advantage normalization	group z-score	$\delta = 10^{-6}$ (Equation (11))
Reward	outcome (Equation (9))	verifiable, length-agnostic

D Halting, Value Head, and Their Phase-1 Pretraining

ReGRAM's halt and value heads use GRAM's ACT and LPRM objectives in Phase 1, providing pretrained stopping and value estimates for Phase 2. The halt head additionally receives the residual feature.

Halt head in Phase 1 (Q-target regression). As in HRM/TRM/GRAM, the halt head outputs $q_\psi(h) = (q^{\mathrm{halt}}, q^{\mathrm{cont}})$ and is trained by a temporal-difference regression with targets $\hat{q}_n^{\mathrm{halt}} = \mathbf{1}[\hat{y}^{(n)} = y]$ and $\hat{q}_n^{\mathrm{cont}} = \max(q_{n+1}^{\mathrm{halt}}, q_{n+1}^{\mathrm{cont}})$, via

$$\mathcal{L}_{\mathrm{ACT}} = \sum_n \Big[(q_n^{\mathrm{halt}} - \hat{q}_n^{\mathrm{halt}})^2 + (q_n^{\mathrm{cont}} - \hat{q}_n^{\mathrm{cont}})^2 \Big]. \tag{19}$$

The change from GRAM: the head consumes $[h_T; \rho_n]$ rather than h_T alone, so the residual feature is already integrated when the head is reinterpreted as the halting policy π_ψ in Phase 2 (we initialize the Phase-2 policy logit as $g_\psi = q^{\mathrm{halt}} - q^{\mathrm{cont}}$). This auxiliary loss flows gradients only into the head, not the recursive core, in Phase 1.

Value head / LPRM. The value head $v_\psi(z_t)$ regresses to the trajectory's final accuracy $r \in [0, 1]$, $\mathcal{L}_{\mathrm{LPRM}} = \sum_t (v_\psi(z_t) - r)^2$, and serves two Phase-2 roles: the shaping potential in Equation (10) and the optional critic baseline. Sharing the head reuses the learned value estimate; the telescoping identity in Section A.4 follows from the fixed potential and its zero terminal value.

Deterministic inference halting. At deployment we halt greedily when $\sigma(g_\psi([h_T; \rho_n])) > \frac{1}{2}$. The rule remains defined across depth budgets under Proposition 4.1; Section 6.2 and Figure 4b evaluate accuracy and calibration transfer. Parallel trajectories may halt at different depths, composing with width-based selection as in GRAM.

E Additional Experiment Setup

E.1 Datasets and Metrics

We follow GRAM's task construction exactly so that the only difference between GRAM and ReGRAM rows is the RL phase.

Sudoku-Extreme (Wang et al., 2025): 9×9 puzzles with minimal clues; 81 tokens, vocab 11. Metric: exact-board accuracy. Difficulty proxy for Section 6.2: number of blanks (81 − clue count).

ARC-AGI-1/2 (Chollet, 2019; Chollet et al., 2025): variable grids padded to 30×30 (900 tokens, vocab 12) with task-specific puzzle embeddings. Metric: exact-output accuracy (pass@1 unless stated). For width experiments we report majority vote and LPRM/best-of-N selection. Difficulty in Section 6.2 is the frozen pre-RL failure rate over 32 GRAM trajectories per instance.

N-Queens (8×8, 10×10) and Graph Coloring (8/10 vertices): constructed as in GRAM (remove $k \in \{5,6,7\}$ / $\{7,8,9\}$ queens; Erdős–Rényi 3-colorable graphs, canonical-form solution enumeration). Metrics: single-sample accuracy / conflict edges, and coverage (unique valid solutions over 20 samples). Difficulty proxy: ground-truth solution multiplicity.

E.2 Baselines and Controls

Published deterministic recursive and GRAM baselines in Table 1 provide external references. Two control conditions test Remark 3.3 empirically: (i) *TRM + naive inference noise* (PTRM-style), which adds randomness at inference without a policy objective; and (ii) *TRM + grafted stochastic head + RL*, which adds a Gaussian head to a pretrained deterministic core and runs the same Phase 2, without Phase-1 ELBO training of that head. These controls test whether generative pretraining improves accuracy and stability beyond the effect of added randomness.

E.3 Measuring Emergence

For Section 6.2 we log, every RL evaluation: mean and distribution of halting depth H, accuracy, and their correlation with the difficulty proxy. We measure the difficulty–depth correlation and its evolution with accuracy under $\lambda_{\text{step}} = 0$. The $\lambda_{\text{step}} > 0$ ablation measures the effect of an explicit computation cost.

E.4 Evaluation Protocol, Seeds, and Statistical Reporting

The evaluation uses the following seed, decoding, and compute-accounting conventions.

Seeds and repeats. Primary ReGRAM, ablation, CSP, and matched-control runs use three seeds. Where shown, $\pm$ denotes the standard deviation across seeds. The difficulty–depth analysis uses held-out per-instance trajectories and reports Spearman correlation.

Evaluation cadence. During Phase 2 we evaluate every fixed number of policy updates (default: every 200 updates), logging accuracy, mean/quantile halting depth, average NFE, group-reward mean/variance, the prior-anchor KL, and the fraction of invalid (masked) rollouts. The anchor KL and invalid fraction are our two early-warning signals for the manifold-drift failure of (Deng et al., 2026): a rising invalid fraction or runaway KL triggers a learning-rate or β_{rl} adjustment.

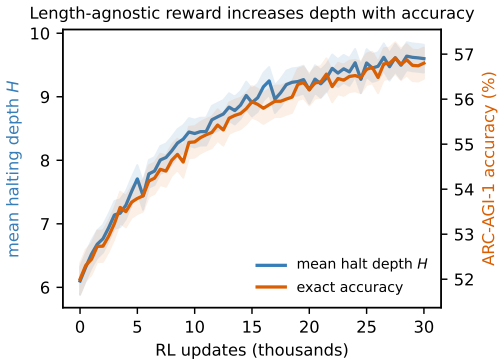
Decoding for evaluation. Evaluation uses deterministic halting ($\sigma(g_\psi) > \frac{1}{2}$). Single-trajectory accuracy, selected-output accuracy from N samples (LPRM or majority vote), and verifier pass@N are distinct metrics. Matched comparisons use the same selector and sampling budget.

ARC augmentation, matched to GRAM. GRAM showed that ARC accuracy depends heavily on the number of train-time augmentations and that inference-time sampling and augmentation are complementary (saturating together). To avoid conflating our width/depth effects with augmentation, we follow GRAM's ablation design: we report (a) the no-augmentation setting, isolating the contribution of RL-driven depth and sampling, and (b) the matched-augmentation setting for comparability with reported GRAM/TRM numbers. The emergence and extrapolation analyses (Section 6.2) use the no-augmentation setting so that any depth–difficulty relationship is a property of the model, not of augmentation-induced diversity.

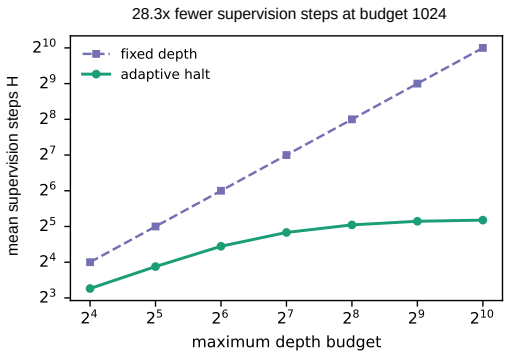
Compute accounting. We distinguish mean supervision depth H from high-level transition evaluations: per-trajectory NFE is TH, and useful group NFE is $T\sum_i H_i$. Padding to a group's longest trajectory instead executes up to $TN\max_i H_i$ evaluations; masking alone does not save this padded work. Hardware efficiency additionally depends on executed work, batching, and wall-clock time.

E.5 Adaptive-Depth and Extrapolation Plots

Figure 2 presents the main difficulty and extrapolation diagnostics. Figure 5 reports the RL training trajectory and mean computation at each inference budget. Computation is expressed in supervision steps per trajectory, H. With $T = 3$, high-level transition evaluations equal $3H$; total group compute sums over the N trajectories. The $28.3\times$ reduction compares supervision-step counts and excludes batching and hardware overhead.



(a) RL training curve: mean depth 6.1 to 9.6, accuracy 52.0% to 56.8%.



(b) Mean computation, in supervision-step units: mean depth saturates at 36.2.

Figure 5: Training dynamics and computation. These panels complement the main difficulty and accuracy diagnostics.

E.6 Multi-Solution Coverage

The component ablation is reported in the main text (Section 6.2); the complete multi-solution table is reproduced here for reference.

Table 6: Multi-solution benchmarks. TRM and GRAM are reported baselines. Accuracy and coverage are percentages; Graph columns report 8/10-vertex results. Coverage uses 20 samples. Coverage losses from GRAM range from 0.5 to 0.9 percentage points.

	N-Queens 8×8		**N-Queens** 10×10		**Graph Coloring (8/10)**	
Method	Acc.	Cov.	Acc.	Cov.	Conflict↓	Cov.
TRM	66.8	36.1	17.5	2.0	109.3/170.7	22.3/6.8
GRAM (Phase 1)	99.7	90.3	89.7	57.5	2.7/3.3	85.8/51.3
ReGRAM	**99.9**	89.8	**93.0**	56.9	**1.8/2.4**	84.9/50.7

F Limitations

The boundedness analysis assumes an invariant contractive basin; on tasks where the latent dynamics do not settle, the residual signal is less informative and halting may revert to budget-bounded behavior. Like GRAM, the sequential deep-supervision schedule limits training throughput, and Phase-2 rollouts add the usual RL sample-efficiency cost on top. Our reward is verifiable-task-bound; extending ReGRAM to domains without a cheap verifier would require a learned reward model and reopens the manifold-stability questions the anchor seeks to mitigate. The evaluation scope is structured puzzles and small models; scaling learned recursive depth to foundation models remains future exploration.