VERILOOP E2: TOWARD VERIFIABLE INTELLIGENCE

Libo Wang*, Anlin Xiang, Haiqi Li, Houde Liu

September 2026

wanglibo@sz.tsinghua.edu.cn

ABSTRACT

VeriLoop E2 is a 27B post-trained model built on Qwen 3.8-27B for code agents, mathematical reasoning, and scientific problem solving. Its core mechanism, VeriLoop-Governed Recurrence (VGR), separates model-side proposal from external verification and persistence: the model generates candidate transitions, while a fixed verification contract admits a transition only when no protected evidence coordinate regresses and at least one strictly improves. This makes VGR particularly well aligned with post-training, because its supervision is defined over task-level state transitions—verified progress, rejected regression or incomparability, and certified completion—rather than isolated next-token prediction. These labels require a model that can already produce meaningful candidate artifacts, together with a stable task identity, an incumbent state, and an executable verifier contract; such conditions naturally arise in downstream trajectories but are generally unavailable during web-scale pretraining, while ordinary domain mid-training can improve distributional familiarity without creating the same-incumbent counterfactual labels required by the protected order. VeriLoop E2 therefore uses post-training to reshape the proposal distribution of an already capable base model toward corrections that are more frequently compatible with externally checkable progress, while keeping the verifier and commit rule outside the gradient path so that learning improves proposal quality without absorbing persistence authority into model weights. The training strategy follows this principle through a structured 1.84M-record corpus spanning software engineering, tool use, mathematical reasoning, scientific reasoning, and verifier-governed trajectories, with capability-specific routing and protected-core replay rather than a single undifferentiated mixture; verified strict-progress actions supply correction supervision, noncommittable candidates provide contrastive signals, and zero-rank artifacts provide final-generation targets under QLoRA-based adaptation. Across the six publicly disclosed general-capability surfaces, every recorded final evaluation loss is lower than its initial value. The resulting checkpoint is evaluated across nine benchmarks covering repository-level software engineering, terminal and agentic execution, mathematics, and scientific reasoning, and is further examined through two evidence-closed scientific reasoning demos executed with the team's internal trial Harness. Our findings indicate that VGR provides a coherent post-training framework for converting externally verified trajectories into learned correction behavior while preserving a strict separation between model reasoning and verification authority, offering a practical systems path toward verifiable intelligence.

Introduction, Related Work, Architecture and Algorithm, Evaluation, Scientific Reasoning Demos, and Conclusions

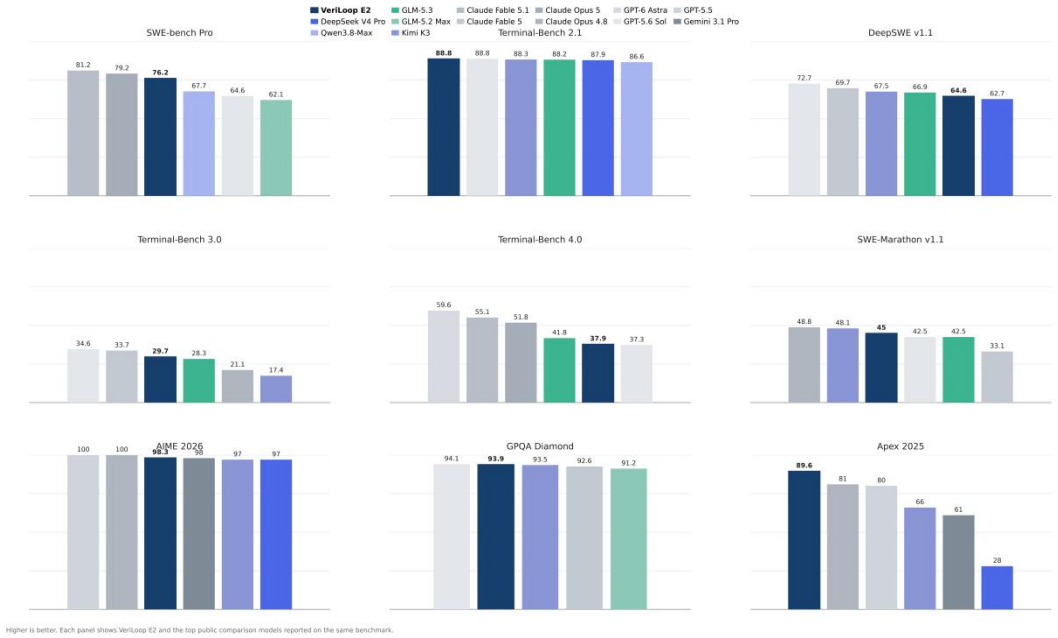


Figure 1 | Cross-Model Benchmark Performance of VeriLoop E2 Across Code, Agentic, Mathematical, and Scientific Reasoning Tasks

1 Introduction

Additional inference computation does not by itself establish progress on a task. Transformer decoders propagate newly computed representations through successive layers, while recurrent variants reuse transformations to increase effective depth [1–3]. Neither an additional layer application nor a stable prediction distribution, by itself, establishes that the resulting output satisfies more of a task's externally checkable requirements. Process supervision and verifier-guided decoding address parts of this problem through progress estimation and executable checks [4, 5]. The narrower problem addressed here is to specify an explicit rule under which a candidate latent state may replace the state currently retained by the system without sacrificing any previously protected evidence. This is a state-management problem for verifiable tasks, rather than a claim that Transformer computation is intrinsically invalid or that all previous verification methods lack commit control.

VeriLoop Coder-E1 approached reliable code generation through an evidence-governed system loop. Its public model card distinguishes the model backend, four narrow-domain PEFT surfaces for ToolSpec, Uncertainty, Rollback, and Evidence Binding, and a private Self-Harness responsible for execution control; its reported benchmark results belong to the complete evaluated system [6]. The institutional release describes the same coupling of generation, tool execution, testing, correction, and regression control [7]. This design leaves a specific opportunity for E2. Behavioral adaptation can help a model cooperate with the Harness, but it does not by itself define a mathematical acceptance order over latent transitions. Likewise, successful orchestration does not establish how much correction capability has been internalized by the model. These are limits of scope and attribution, not contradictions in the E1 evidence-governed principle.

We develop VeriLoop-Governed Recurrence (VGR), the evidence-governed recursive mechanism underlying the VeriLoop E2 research program. VGR separates three authorities: a learned proposer generates a bounded latent displacement; a fixed decoder materializes its task artifact; and an external verifier evaluates that artifact against an immutable set of protected obligations. A deterministic controller commits the candidate only when every protected evidence coordinate is no worse and at least one is strictly better. A rejected proposal may inform the next attempt, but it cannot overwrite the retained artifact, latent state, or verification record. This extends evidence-governed correction from an agent workflow to an explicit model-state interface. This separation provides the operational meaning of the report's title phrase "toward verifiable intelligence": the

objective is not to claim that every generated statement is formally proven, but to make consequential state changes and retained claims auditable, rejectable, and rollback-safe under external evidence.

The distinction from changes to information flow is substantive. Attention Residuals changes how a layer aggregates preceding representations [8]. VGR governs whether a proposed state transition may be retained after its observable consequences have been checked. The two mechanisms address different questions; their relative empirical importance cannot be inferred from architectural descriptions. VGR does not require replacement of attention or feed-forward sublayers. It requires an exposed latent interface and a decoder that actually consumes the proposed latent, together with an independently enforced verification contract.

The same acceptance relation also defines supervision for post-training. Verified improvements can supply correction targets; non-improving or regressing candidates can supply comparison examples; and zero-rank artifacts can supply final-answer targets. This makes VGR compatible with adaptation of a pretrained model without differentiating through a compiler, test runner, or discrete commit decision. Learning is intended to increase the frequency and usefulness of admissible proposals. The external controller remains responsible for enforcing the acceptance rule at deployment. Learning from VGR trajectories alone does not install a latent recurrence module or transfer the controller's guarantees into the model weights.

This report develops three contributions. First, it specifies a protected partial order with stable obligation identities, preventing aggregate score improvements from concealing a regression. Second, it gives an atomic state transition that binds a candidate latent to its decoded artifact and verification result, with exact retention on rejection. Third, it defines a low-rank proposal operator and a verifier-consistent supervision interface, connecting the architectural mechanism to post-training without conflating learned behavior with formal authority. The resulting guarantee is precise: committed evidence ranks improve monotonically and the number of strict commits is finite. Reaching a zero rank, semantic correctness beyond the verifier's scope, and downstream model gains require separate conditions or experiments.

The present report contains the introduction, related work, architectural and algorithmic specification, the finalized benchmark campaign, post-training optimization measurements, and two scientific reasoning demos. VeriLoop E2 is built by post-training Qwen 3.8-27B; the untouched Qwen 3.8-27B checkpoint remains a separate baseline identity for controlled evaluation. This establishes model lineage for the benchmark campaign without, by itself, proving a causal gain over the raw base under matched conditions. The VGR reference specification remains backbone-independent, while the latent VGR realization specified in Section 3 is a distinct integration layer whose end-to-end implementation must be established separately. This separation prevents three attribution errors: model-side proposal, abstraction, and diagnosis belong to the bound E2 checkpoint; external verification, rollback, and certification belong to the Harness and verifier contract; and improvement caused specifically by post-training is claimed only where matched empirical evidence supports it. Section 5 therefore reports benchmark outcomes and public post-training measurements under this attribution contract, followed by verifier-closed research trajectories produced by the VeriLoop E2 checkpoint together with the team's internal trial Harness at the artifact/system level.

2 Related Work

2.1 Transformer depth and state evolution

The original Transformer establishes attention-based sequence modeling with residual state propagation [1]. Universal Transformers share a transformation across depth and introduce adaptive per-position halting [2]. Recurrent-depth language models further organize computation into input processing, a shared recurrent core, and output decoding, allowing inference depth to vary independently of the number of distinct parameters [3]. These methods provide the computational substrate on which additional latent work can occur. An exit criterion based on internal stabilization answers when to stop that work; it is not an external certificate of task progress.

Mixture-of-Depths assigns computation selectively across tokens through routing under a computation budget [9]. Attention Residuals makes depth-wise aggregation input-dependent [8].

Source-Centered State Evolution constructs a reference-preserving recurrence: its designated anchor is invariant, while nonzero deviations remain computationally active [10]. These are complementary controls over where computation occurs and how representations evolve. An anchor fixed point is a property of a transition map, whereas VGR's protected descent is a property of externally evaluated artifacts associated with accepted transitions. Neither property implies the other.

We use this literature to motivate an exposed state-update interface, not as a proof of VGR's acceptance rule. The rule and its guarantees follow from the definitions in Section 3. In particular, VGR does not assume that an arbitrary pretrained decoder will respond usefully to every bounded hidden-state perturbation. That issue belongs to proposal reachability and integration, rather than to the correctness of the protected comparison.

2.2 Progress supervision and verifier-guided generation

Setlur et al. define process progress through the change in the probability of eventual correctness under a prover policy and train Process Advantage Verifiers to estimate it [4]. This connects intermediate supervision to future task success, rather than to static judgments of a reasoning step. VGR shares the emphasis on transitions, but uses an observable vector of remaining obligations to authorize commits. A learned scalar advantage can help prioritize computation; it cannot substitute for coordinate-wise preservation when the application explicitly protects several obligations.

Decoding Time Verification interleaves generation with deterministic checks at structural boundaries and uses commitment and rollback to control generated prefixes [5]. Its controller already enforces a persistence boundary within decoding. VGR specifies a different controlled object: a latent–artifact–verification bundle, with a relative strict-progress test over fixed evidence coordinates. The distinction concerns the binding and acceptance semantics. A latent-level realization therefore requires a decoder that consumes the controlled state; a token-level implementation of the acceptance rule alone does not establish that integration.

STaR shows how selecting successful reasoning traces and fine-tuning on them can improve a generator [11]. This supports a route from verified outputs to post-training, while leaving the verification and generalization questions separate. VGR additionally distinguishes intermediate strict progress from certified completion and requires preference labels to agree with the runtime acceptance predicate. These distinctions matter when an apparently better repair breaks a previously passing test, or when a useful partial correction is not yet a complete solution.

2.3 Parameter-efficient adaptation and preference objectives

LoRA trains low-rank weight updates while keeping pretrained weights frozen [12]. QLoRA reduces adaptation memory by combining a frozen quantized backbone with trainable adapters [13]. These techniques make an evidence-conditioned proposer practical to adapt; they do not supply an evidence order or an external verifier. The low-rank latent operator specified below is also distinct from a LoRA weight update: one proposes an activation displacement, whereas the other changes the trainable parameterization used to generate proposals.

Direct Preference Optimization relates preference learning to a reference-policy-constrained objective [14]. The executable VGR specification instead provides a margin-based pairwise logistic loss on accepted and rejected sequence log-probabilities. It has no reference-policy ratio and is not DPO. Its role is to favor verified progress examples, while the discrete protected comparison retains runtime authority. This distinction prevents a familiar training label from obscuring the actual objective being optimized.

2.4 From evidence-governed systems to controlled state transitions

E1's documented division between model-side behavioral surfaces and Harness-side governance supplies the immediate system lineage [6]. VGR sharpens that division into a mathematical interface: the proposer has generation authority, and the controller has persistence authority. Relative to the research above, the proposed combination is a fixed protected order, an explicit latent-to-artifact binding, atomic retention on rejection, and supervision derived from that same order. Each component has antecedents in learning, verification, or state management. The technical contribution is their specific composition and the consequences proved for it, rather than a claim to have originated recurrence, verification, low-rank adaptation, or monotone reasoning.

3 Architecture and Algorithm

3.1 Scope and computational boundary

VGR controls a sequence of task-level proposal attempts. The attempt index t is neither a token position nor a Transformer layer index. For one task, let $H_t \in \mathbb{R}^{L \times d}$ denote the retained latent state, where L is its sequence length and d its hidden width. A proposed displacement has the same shape. Candidate artifacts may contain variable-length output sequences even when the controlled latent interface has a fixed shape within a run.

Figure 3 shows the ordinary decoder computation that motivates this interface. It is a generic Transformer schematic, not a claim about the exact attention pattern, normalization, or cache layout of the target Qwen checkpoint. Figure 4 isolates the absence of a task-level verification gate in that baseline computation. Its missing-component annotations refer to the illustrated baseline, not to every prior verifier-augmented system.

Two realizations must be distinguished. In the **latent realization**, the model exposes initialization, proposal, and decoding interfaces, and the decoder consumes the exact candidate latent constructed by the controller. In the **trajectory-supervised realization**, a conventional autoregressive model is adapted using externally verified repair records. The latter is a valid post-training use of VGR supervision, but does not establish that the former has been integrated. Section 3 specifies the latent mechanism and the common supervision interface; it does not report a completed architectural modification of Qwen.

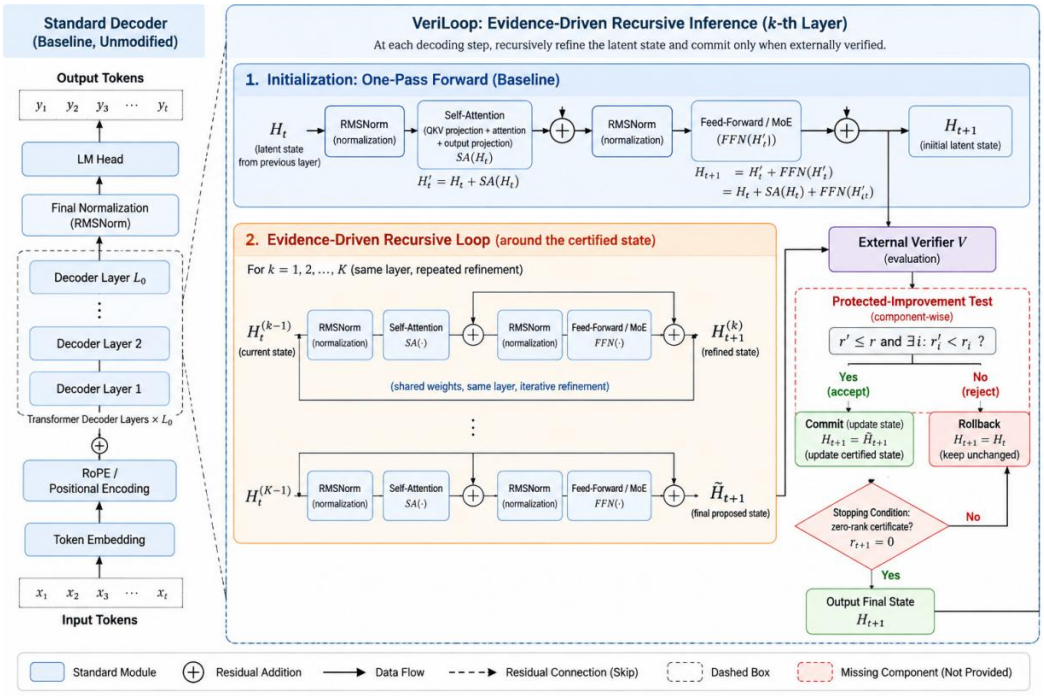


Figure 2 | System-level overview of veriloop-governed recursive inference.

As shown in Figure 2, the latent realization can be understood as a baseline initialization followed by a sequence of task-level proposal attempts around the currently retained state. The ordinary decoder first produces a baseline latent state, which is decoded and externally verified before any recursive proposal is considered. At each subsequent attempt, VGR starts from the retained state, constructs a bounded candidate update, decodes the resulting candidate through the fixed decoder, and submits the bound artifact to the external verifier. The protected comparison then determines whether the complete candidate bundle is atomically committed or whether the incumbent bundle is preserved unchanged. In Figure 2, k is only a local schematic counter for successive proposal attempts; the formal development uses t as the attempt index throughout. Neither the schematic k nor the formal t denotes a Transformer layer index or token position. A zero protected rank

terminates the run with certified completion, whereas rejection preserves the incumbent bundle and allows only the rejected displacement to influence subsequent exploration.

3.2 Task contract and notation

Let x be a task and id_x its nonempty identifier. A canonical digest d_x=Digest(x) binds its contents. During one run, the model, verifier, and evidence compiler have fixed identities:

$$\mathscr{R}=(d_M,\iota_V,d_V,d_\Gamma). \tag{1}$$

Here d_M is the model contract digest; ι_V is the verifier identifier; d_V is the verifier contract digest; and d_Γ is the evidence compiler contract digest. The model contract must cover the active parameters and relevant decoding configuration. A digest records identity; it does not itself validate the semantics of the object being hashed.

The verifier contract specifies an ordered obligation list O=$(o_1,\ldots,o_m)$, with positive integer size m and distinct component identifiers I=$(i_1,\ldots,i_m)$. Each obligation binds a checked object, an evaluation rule, and any applicability or execution-failure policy. Both identities and meanings remain fixed within a run. Different tasks may use different contracts; their rank vectors are not compared unless the obligation semantics have been aligned explicitly.

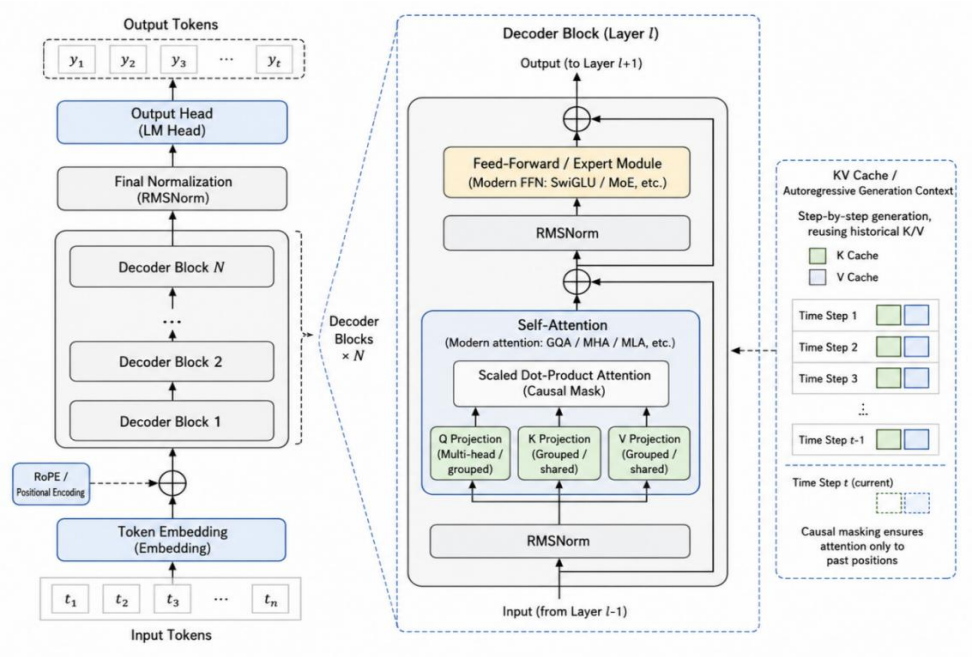


Figure 3 | Generic autoregressive decoder and KV-cache path. Layer computation produces representations and output tokens; this schematic does not assign external certification to them.

Core notation used to follow the VGR state-transition mechanism is summarized below. Appendix A provides the complete notation and symbol glossary, including local-scope symbols, operators, domains, and typographic conventions.

Symbol / operator	Precise definition, domain, and scope
x; idx; d_x	x is the task; idx is its nonempty task identifier; d_x = Digest(x) is the canonical task digest.
$\mathscr{R}$; d_M; ι_V; d_V; d_Γ	$\mathscr{R}$ is the fixed run contract. d_M is the model-contract digest, ι_V the verifier identifier, d_V the verifier-contract digest, and d_Γ the evidence-compiler-contract digest.
$I = (i_1, \ldots, i_m)$	I is the ordered list of distinct obligation identifiers; i_j identifies obligation o_j.
m	$m \in \mathbb{N}$ is the number of protected obligation coordinates in the fixed schema.

Symbol / operator	Precise definition, domain, and scope
t	$t \in \{0,1,2,\ldots\}$ is the formal task-level proposal-attempt index used throughout the mathematical development.
A_x	A_x is the fixed task-conditioned source anchor for the run.
H_0; H_t; H_t'	H_0 is the baseline latent state; H_t is the retained latent at attempt t; H_t' is the candidate latent constructed for that attempt. Each lies in $\mathbb{R}^{L\times d}$ within a run.
Δ_t	$\Delta_t \in \mathbb{R}^{L\times d}$ is the proposer-supplied latent displacement for attempt t.
$F_{\theta,\phi}$	$F_{\theta,\phi}$ is the learned proposal function that maps the task, anchor, retained state, rejected-displacement memory, evidence feature, and attempt index to Δ_t.
ρ; τ_{lat}	$\rho > 0$ is the controller legality radius for each latent-position displacement; $\tau_{\text{lat}} \geq 0$ is the declared numerical tolerance used only in that legality check.
θ; ϕ	θ denotes backbone/model parameters for the latent realization; ϕ denotes trainable adaptation parameters. They are held fixed within a certified run snapshot.
$D_{\theta,\phi}$	$D_{\theta,\phi}$ is the fixed decoder used within a run; it materializes task artifacts from the controlled latent state.
V	V is the external verifier. It receives the task and a bound artifact and returns a structured verification result under the fixed verifier contract.
r; r_t; r_t'	$r \in \mathbb{N}_0^m$ is a complete protected evidence-rank vector; r_t and r_t' are the incumbent and candidate ranks at attempt t. Smaller coordinates mean fewer remaining violations under the fixed obligation semantics.
$\preccurlyeq_P$; $\prec_P$; $\not\prec_P$	$\preccurlyeq_P$ is weak protected dominance (no coordinate worsens); $\prec_P$ is strict protected dominance ($\preccurlyeq_P$ plus at least one strict improvement); $\not\prec_P$ means the strict protected relation does not hold.
χ_t	$\chi_t = 1\{r_t' \prec_P r_t\}$ is the commit indicator: 1 authorizes atomic commitment and 0 requires retention of the incumbent bundle.
B_t; B_t'	$B_t = (a_t, H_t, v_t, \beta_t)$ is the complete retained CertifiedBundle; B_t' is the candidate bundle formed from a_t', H_t', v_t' and its recomputed digest.
Φ; Σ / $\sum$	Φ is the integer evidence potential. The symbols Σ and $\sum$ denote summation. In the definition of Φ, the summation runs over protected-coordinate index j through 1 through m. In Equation (26), the summation runs over proposal-attempt index t across the finite run and counts strict commits through χ_t. Potential descent is only a proof/diagnostic consequence and never replaces the coordinate-wise commit predicate.
Γ	Γ is the fixed evidence compiler. It maps incumbent/candidate verification information and decision metadata to a bounded model-visible feature; it has no commit authority.
e_t	$e_t \in [-1,1]^q$ is the model-visible neural evidence feature at attempt t.

The symbol B_t retains the implementation name CertifiedBundle. At a nonzero rank, it means a bundle admitted to the protected state chain with a recorded verification result; it does **not** mean that the task has been solved. The separate Boolean field certified denotes zero-rank completion. This terminological distinction is essential for interpreting the baseline and intermediate states.

3.3 Initialization and fixed decoding

Initialization returns a source anchor and a baseline latent:

$$(A_x, H_0) = \text{Initialize}_{\theta,\phi}(x), \qquad E_0 = 0, \quad s_0 = 0. \tag{2}$$

The source anchor is a fixed task-conditioned representation for this run. The baseline latent is produced by the model's ordinary initialization path. VGR does not assume that the anchor is a fixed point of the recurrent operator, or that $A_x = H_0$. The reference controller takes private copies so that subsequent proposal calls cannot modify these objects through tensor aliases.

The controller decodes and verifies the baseline before the first proposal:

$$\begin{aligned} a_0 &= \mathrm{Bind}\quad \big(D_{\theta,\phi}(x,H_0),H_0,x,\mathscr{R}\big),\\ v_0 &= V(x,a_0). \end{aligned} \tag{3}$$

Here $D_{\theta,\phi}$ is the fixed decoder for the current run, V is the external verifier, and Bind constructs an artifact identity from the decoded output, witness, metadata, latent digest, task digest, and model contract. The subscript ϕ permits an already adapted decoder; its parameters are frozen during a run even if they were learned earlier. The witness is a machine-checkable object associated with the output, not necessarily a formal proof.

The initial evidence feature is

$$e_0=\Gamma(v_0,v_0,0,-1)\in[-1,1]^q. \tag{4}$$

The third argument records that initialization is not a candidate commit, and the index -1 distinguishes it from attempt zero. If the baseline has zero rank, the controller returns certified completion immediately after this initialization. Otherwise, the first proposal is already conditioned on externally obtained evidence.

Deterministic decoding fixes the output rule, stop conditions, tie-breaking, and relevant execution settings. A repeated task–model–latent key must produce the same artifact identity. The verifier likewise must return the same verification result for a repeated candidate identity. Inconsistent repeats, changed contracts, malformed outputs, or mutation of protected inputs raise a contract error. Such an error is not an ordinary failed repair and must not produce a success certificate.

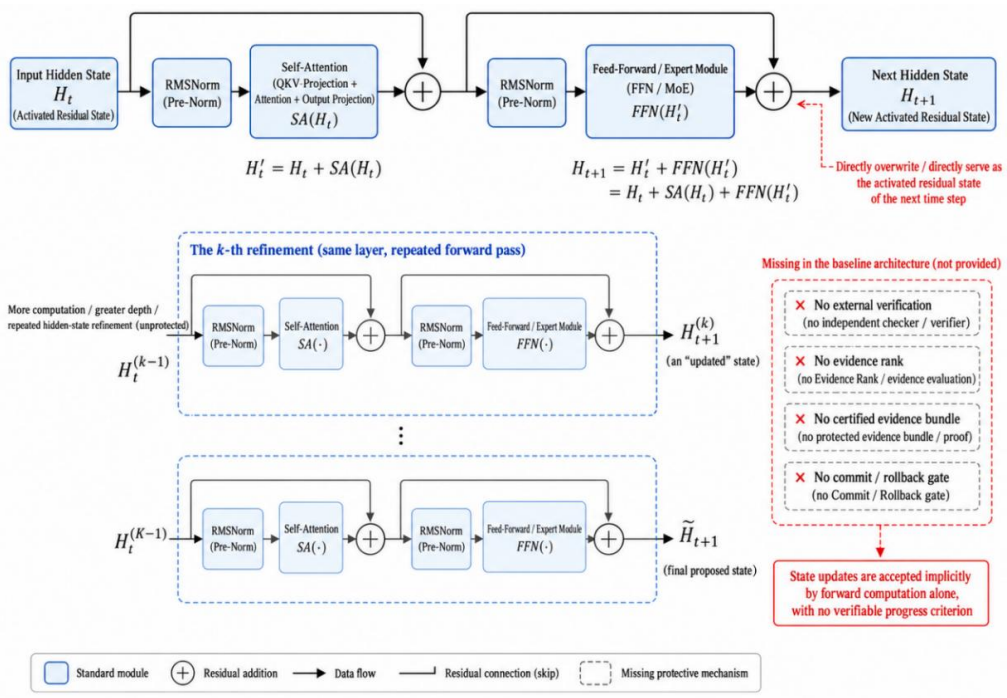


Figure 4 | Baseline residual propagation and repeated-depth computation. A new activation becomes an input to subsequent computation without an artifact-level protected-progress decision in the illustrated path.

3.4 Protected evidence and the commit order

For a bound artifact a, the verifier returns

$$v=(r,c,\iota_V,d_V,\zeta),\qquad r=(r_1,\ldots,r_m)\in\mathbb{N}_0^m. \tag{5}$$

The vector r is the complete protected evidence rank, c is the completion flag, and ζ is a structured diagnostic receipt. Smaller coordinates represent fewer remaining violations under the declared rules. For a binary test obligation, zero can mean pass and one fail. Integer counts are also

permitted, but a count protects only its specified meaning: the total number of failing tests does not preserve the identity of each previously passing test.

Two ranks can be compared only under the same ordered schema and verifier contract. With this precondition, define weak and strict protected dominance by

$$r' \preccurlyeq_P r \Leftrightarrow \left(\forall j \in \{1,\ldots,m\},\ r'_j \le r_j\right), \tag{6}$$

$$r' \prec_P r \Leftrightarrow r' \preccurlyeq_P r \wedge \left(\exists j \in \{1,\ldots,m\},\ r'_j < r_j\right). \tag{7}$$

The subscript P denotes the fixed protected order. A mismatched schema causes an error before comparison rather than a permissive interpretation of the same vector length. The unique commit indicator is

$$\chi_t = \mathbf{1}\{r'_t \prec_P r_t\}. \tag{8}$$

The indicator $\mathbf{1}\{\cdot\}$ is one when its predicate is true and zero otherwise. In particular, a learned score, a favorable receipt summary, or a lower aggregate error count cannot override a zero value of χ_t.

Figure 5 expands this compact decision matrix into the full protected-order semantics, including fixed-schema comparability, the unique commit predicate, atomic state retention, and the finite strict-commit consequence.

Retained rank	Candidate rank	Decision	Reason
(3,2,1,0)	(3,1,1,0)	Commit	One coordinate improves and none worsens
(0,2)	(1,0)	Reject	The total falls, but the first obligation regresses
(2,1,0)	(2,1,0)	Reject	No strict progress
(1,1)	(0,2)	Reject	The candidates are incomparable under the protected order

An obligation must not disappear because a candidate deletes the function or file it concerns. Missing or unexecutable checks receive the failure treatment declared in the contract; malformed results fail closed. Applicability can depend on task conditions, but its rule must be specified in advance and must not let a candidate erase an unresolved requirement. These requirements give the coordinates stable semantics, without which a coordinate-wise inequality would have little evidential meaning.

3.5 Proposal authority and latent construction

The learned proposer receives the task and private copies of the current neural inputs:

$$\Delta_t = F_{\theta,\phi}(x, A_x, H_t, E_t, e_t, t). \tag{9}$$

The controller checks shape, floating-point representation, device, finite values, and the per-position displacement bound

$$\max_{1 \le \ell \le L} \| \Delta_{t,\ell} \|_2 \le \rho + \tau_{\text{lat}}. \tag{10}$$

Here ℓ is a latent position, $\|\cdot\|_2$ is Euclidean norm, $\rho > 0$ is the allowed exploration radius, and $\tau_{\text{lat}} \ge 0$ absorbs declared numerical tolerance. This is a bound on each displacement, not a global bound on the retained latent trajectory. After validation, the controller alone constructs

$$H'_t = H_t + \Delta_t. \tag{11}$$

The candidate latent is then decoded into output y'_t, witness w'_t, and metadata ω'_t. The binding operation materializes their common identity:

$$\begin{aligned} \left(y'_t, w'_t, \omega'_t\right) &= D_{\theta,\phi}\left(x, H'_t\right), \\ a'_t &= \text{Bind}\left(y'_t, w'_t, \omega'_t, H'_t, x, \mathscr{R}\right). \end{aligned} \tag{12}$$

The verifier evaluates $v'_t = V\left(x, a'_t\right)$ and exposes its rank r'_t. It receives the task and artifact, not the controller's latent tensor. The latent is checked indirectly through its decoded consequences and

their binding. This neither proves that a hidden representation is intrinsically correct nor evaluates all outputs that might be decoded from it under a different rule.

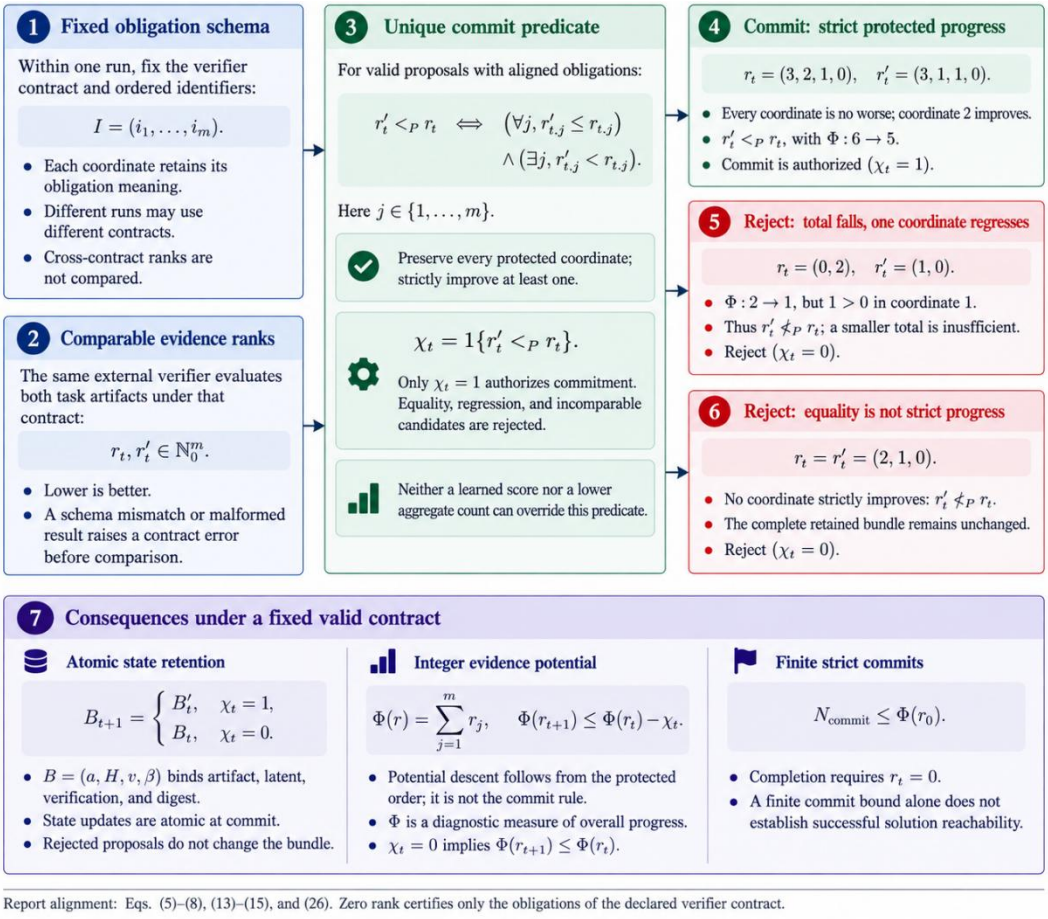


Figure 5 | Protected evidence order and atomic retention under a fixed valid contract. Candidate ranks are comparable only under the same obligation schema; commitment requires coordinate-wise non-regression with at least one strict improvement. Aggregate potential descent is a consequence, not an alternative commit criterion.

3.6 Atomic commit and rejection

The protected unit is the bundle

$$B_t=(a_t,H_t,v_t,\beta_t), \tag{13}$$

$$\beta_t=\text{Digest}\ (\text{id}_x,d_x,\text{Digest}(A_x),a_t,\text{Digest}(H_t),v_t,\mathscr{R}). \tag{14}$$

The canonical representation preserves types and, for tensors, includes shape, dtype, and contiguous bytes. The reference implementation uses SHA-256. This is computational binding under collision resistance, not a cryptographic signature, a proof of provenance, or an information-theoretic guarantee. Object hashes also do not replace process isolation against arbitrary hostile code.

Let B'_t be the candidate bundle formed from a'_t,H'_t,v'_t and its recomputed binding digest. The complete update is

$$B_{t+1}=\begin{cases}B'_t, & \chi_t=1,\\ B_t, & \chi_t=0,\end{cases} \qquad E_{t+1}=\begin{cases}0, & \chi_t=1,\\ \Delta_t, & \chi_t=0.\end{cases} \tag{15}$$

Thus rejection preserves the whole bundle, rather than merely restoring its rank. A rejected displacement is retained only in the exploratory variable E_{t+1}. On a successful commit, the new latent becomes the proposal origin and that displacement memory is cleared. The retained and

exploratory paths are distinct, which allows unsuccessful attempts to influence future proposals without gaining persistence authority.

Figure 6 summarizes these interfaces. The reference implementation additionally checks bundle binding and contract identity around calls. These checks detect inconsistencies in the supported execution model. They do not prove that a test suite is complete or that an external tool reports the truth.

3.7 Evidence feedback and rejected exploration

Evidence is compiled from the actual attempted transition before the retained verification object is replaced:

$$e_{t+1}=\Gamma\left(v_t,v'_t,\chi_t,t\right). \tag{16}$$

Both arguments remain the pre-attempt incumbent result and the current candidate result, including when the candidate is accepted. Using the new result in both positions would erase the measured transition. The compiler has no commit authority. Its output may summarize a large rank vector for neural conditioning, while the complete vector remains available to the protected comparator.

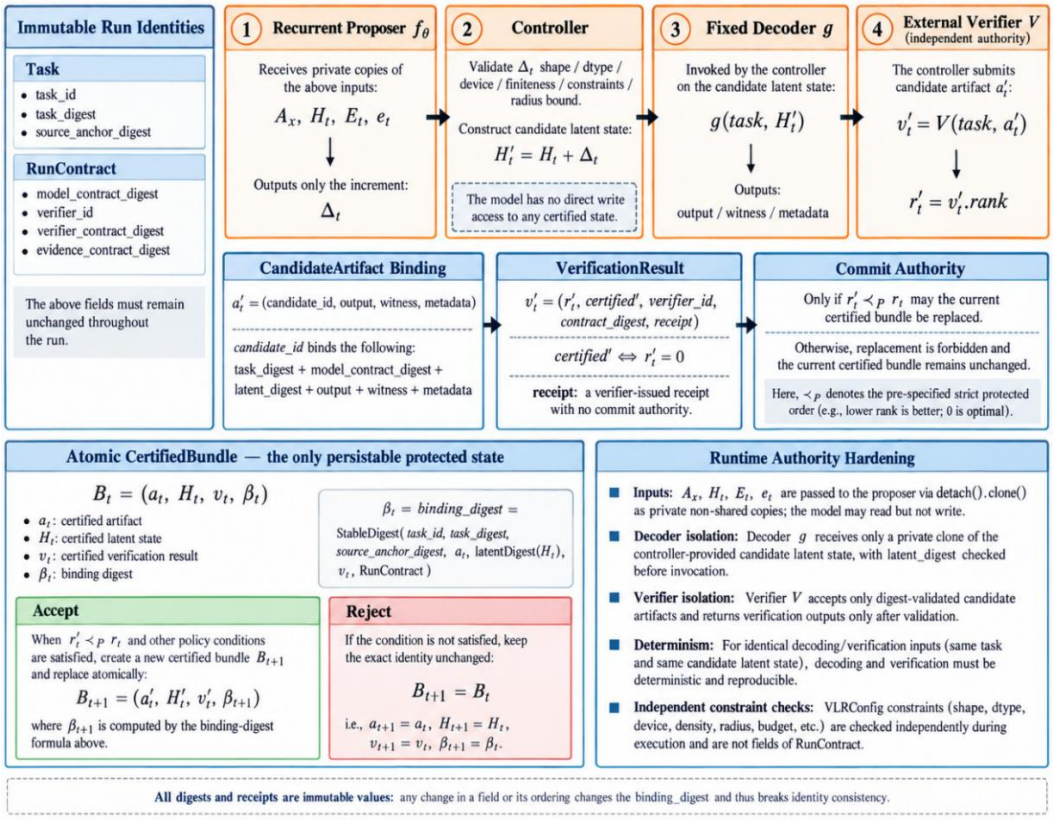


Figure 6 | Binding and authority separation in the VGR reference specification. The decoder is denoted by g in the artwork and by D in this report. CertifiedBundle denotes the retained verified record; only a zero rank means completed certification. Other policy conditions in the artwork are contract-legality checks, not an alternative commit score.

Figure 7 assembles the complete attempt path and makes the ordering constraint explicit: feedback is compiled from the incumbent and candidate verification results before any accepted candidate replaces the retained bundle.

The reference receipt-augmented compiler uses a ten-dimensional rank summary. Define $\Phi(r)=r_1+\cdots+r_m$, a positive feature scale κ, $f(u)=\tanh(\log(1+u)/\kappa)$ for $u\geq 0$, and $b(z)=2\mathbf{1}\{z\}-1$. Let

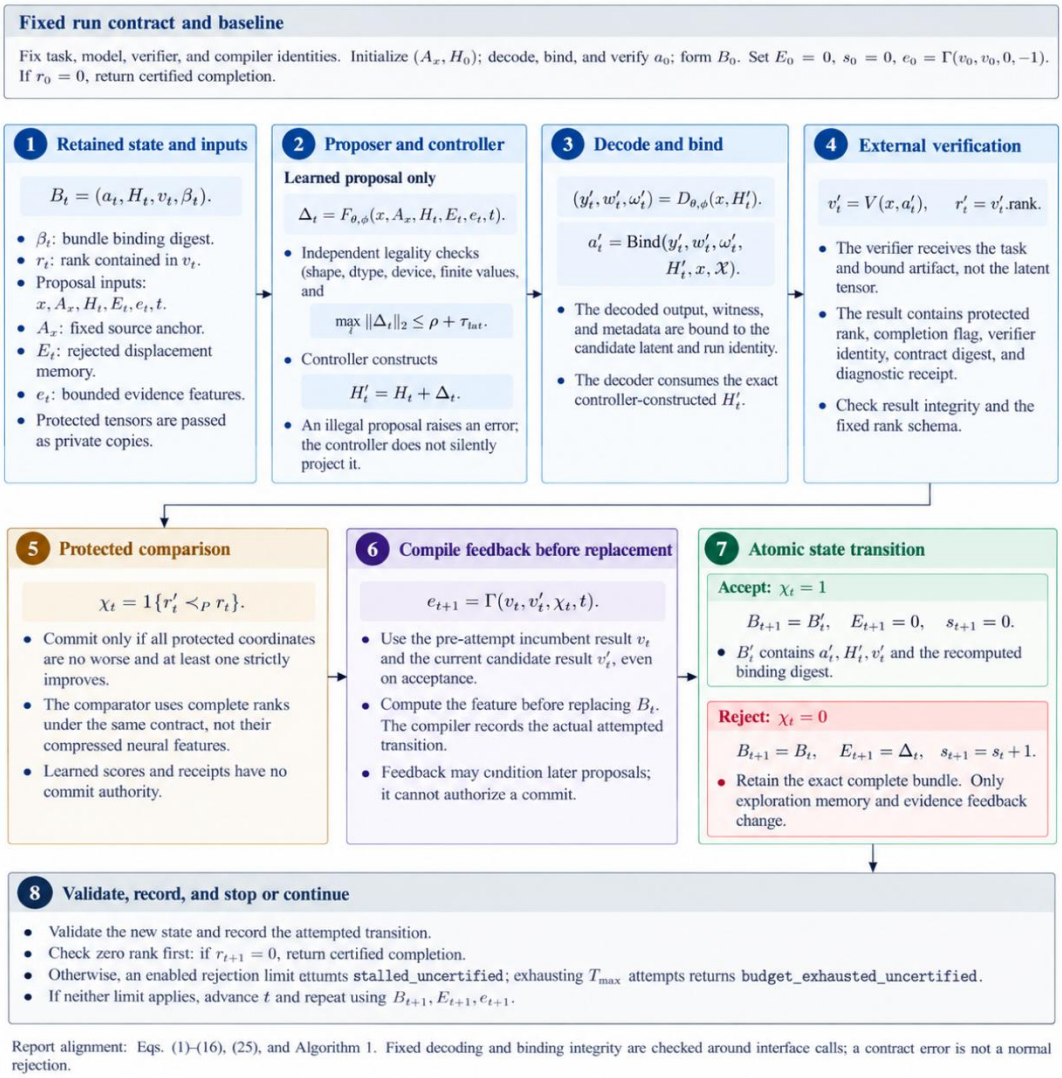
$p_<, p_>, p_=$ be the fractions of coordinates that respectively improve, worsen, or remain unchanged from r_t to r'_t. Then

$$\begin{aligned} \mathrm{h}_{t,1:3} &= \Big(f(\Phi(r_t)), f\big(\Phi(r'_t)\big), \tanh\Big(\big(\Phi(r_t) - \Phi(r'_t)\big)/\kappa\Big)\Big), \\ \mathrm{h}_{t,4:6} &= (p_<, p_>, p_=), \\ \mathrm{h}_{t,7:10} &= \Big(b(\chi_t{=}1), b\big(r'_t{=}0\big), b(r_t{=}0), \tanh(t/16)\Big). \end{aligned} \tag{17}$$

Here $\mathrm{h}_{t,a:b}$ denotes the inclusive component slice from a through b; concatenating the three slices defines $\mathrm{h}_t \in \mathbb{R}^{10}$. If ψ is a fixed receipt encoder of width q_ζ, the feature vector is

$$e_{t+1} = \Big[\mathrm{h}_t; \psi\big(\zeta_t, \zeta'_t, \chi_t, t\big); 0_{q-10-q_\zeta}\Big], \qquad q \geq 10 + q_\zeta. \tag{18}$$

Semicolons denote concatenation and 0_k a vector of k zeros. Every feature must be finite and bounded in $[-1,1]$. The receipt encoder, its dimensions, and its layout are fixed by the compiler contract. Sanitized diagnostics may identify failed checks, error classes, and locations. Hidden expected outputs, reference patches, or fields that directly reveal the correct next answer do not belong in model-visible feedback. Rich permitted diagnostics can also be supplied as text; a compact numeric summary need not be the model's only observation channel.



Report alignment: Eqs. (1)–(16), (25), and Algorithm 1. Fixed decoding and binding integrity are checked around interface calls; a contract error is not a normal rejection.

Figure 7 | Verification and atomic-commit loop under a fixed run contract. Proposal, decode-and-bind, external verification, protected comparison, pre-replacement evidence compilation, and state retention are separate operations. Only $\chi_t = 1$ authorizes commitment, and zero rank is checked before resource stopping.

The rank summary is intentionally lossy and is not described as an equivalent verifier. Its scaling constants and feature width are implementation choices. The formal acceptance theorem does not depend on this particular compiler, and compression must never be applied to the rank vector used for commitment.

3.8 Low-rank latent proposal operator

The reference operator provides a concrete trainable implementation of $F_{\theta,\phi}$. It is an optional proposal parameterization; the controller can accept other proposal functions satisfying the same interface. For clarity, the equations use row-major latent matrices and suppress the batch dimension. Let $r_a \le d$ be the bottleneck width. Choose $W_s, W_a \in \mathbb{R}^{d \times r_a}$, $W_e \in \mathbb{R}^{q \times r_a}$, and $W_u \in \mathbb{R}^{r_a \times d}$, all without biases. With e_t represented as a column vector, form

$$Z_t = \mathrm{LN}_s(H_t + E_t) W_s + \mathrm{LN}_a(A_x) W_a + \mathbf{1}_L \left(e_t^{\top} W_e \right). \tag{19}$$

The vector $\mathbf{1}_L$ contains L ones and broadcasts the evidence projection across positions. The two outer normalizations are affine-free. Thus $Z_t \in \mathbb{R}^{L \times r_a}$ combines the explored state, the source anchor, and the evidence signal without granting any of them commit authority.

The bottleneck mixer and hidden-width update are

$$T_t = Z_t + \mathscr{M}(Z_t), \qquad U_t = \tanh(T_t W_u), \qquad \gamma = \sigma(g). \tag{20}$$

Here $\sigma(g) = (1 + e^{-g})^{-1}$ is a scalar sigmoid gate and g is trainable. The mixer $\mathscr{M}$ applies affine LayerNorm, a linear map from r_a to $2r_a$, SiLU, optional dropout, and a linear map back to r_a. Unlike the two outer normalizations, its LayerNorm has trainable affine parameters. The reference initializes $g = -4$ and uses small nonzero output-projection weights, giving a small initial update without structurally disconnecting its gradient path.

For a vector $z \in \mathbb{R}^d$, define projection onto a closed Euclidean ball of radius $M_{\mathrm{op}} > 0$ by

$$\Pi_{M_{\mathrm{op}}}(z) = \begin{cases} z, & \| z \|_2 \le M_{\mathrm{op}}, \\ M_{\mathrm{op}} z / \| z \|_2, & \| z \|_2 > M_{\mathrm{op}}. \end{cases} \tag{21}$$

The projection acts independently at each latent position. The implementation uses a small positive denominator floor to avoid division by zero. Its intended use requires a finite positive radius and an appropriate numerical floor; these are proposal-operator configuration requirements, not evidence predicates.

With step size $\eta > 0$, the learned displacement mean is

$$\mu_t = \Pi_{M_{\mathrm{op}}} \quad (E_t + \eta \gamma U_t). \tag{22}$$

If optional exploration noise ξ_t is supplied with scale $\sigma_\xi \ge 0$, the final proposal is

$$\Delta_t = \begin{cases} \mu_t, & \text{no perturbation}, \\ \Pi_{M_{\mathrm{op}}}(\mu_t + \sigma_\xi \xi_t), & \text{otherwise}. \end{cases} \tag{23}$$

The noise tensor has the same shape as μ_t and finite entries. A Gaussian distribution is one possible choice, not a condition of the controller theorem. The internal projection radius M_{op} and external legality radius ρ are distinct. Choosing $M_{\mathrm{op}} \le \rho$ is convenient; the formal controller independently checks the actual result using Equation (10). It does not silently repair an illegal proposal by adding another projection.

The exact trainable parameter count for this reference module is

$$N_{\mathrm{train}} = 3 d r_a + q r_a + 4 r_a^2 + 5 r_a + 1. \tag{24}$$

The first term counts state, anchor, and output projections; the second counts evidence projection; the quadratic term counts the two mixer weight matrices; $5r_a$ counts the mixer's normalization and linear biases; and the final scalar is g. This count excludes backbone parameters, any additional LoRA modules, and task-specific decoder components. The operator changes an activation proposal, not the protected comparison.

Figure 8 summarizes this operator without changing its authority boundary: the module proposes a bounded activation displacement, while the controller still checks the realized proposal independently under Equation (10).

3.9 Complete recurrence and stopping

Algorithm 1 states the execution order. Its notation is consistent with the preceding equations and the version 2.4.0 reference controller. Integrity checks surround interface calls even where omitted from the compact presentation. No branch delegates acceptance to the proposer or the evidence compiler.

The consecutive-rejection counter obeys

$$s_{t+1}=\begin{cases}0, & \chi_t=1,\\ s_t+1, & \chi_t=0.\end{cases} \qquad c_t=\mathbf{1}\{r_t=0\}. \tag{25}$$

$T_{\max}$ is a finite nonnegative attempt budget; the optional $S_{\max}$ is a positive integer. Certification is checked before resource stopping. Assuming each component invocation terminates, these limits bound the number of attempts. A wall-clock guarantee additionally requires timeouts or bounded execution for decoding, verification, and other calls. A stalled or budget-exhausted result retains the best state admitted by the protected order but carries no completion certificate.

An audit trace records the attempt index, incumbent and candidate identities, pre-attempt rank, candidate rank, post-attempt rank, acceptance decision, completion flag, and maximum displacement norm. This distinguishes an attempted improvement from a committed one and supports replay of the decision chain. The trace is evidence about that run; it is not a benchmark score.

Algorithm 1 VGR with protected atomic commitment

Inputs: task x, task identifier id_x, model interfaces $F_{\theta,\phi}, D_{\theta,\phi}$, $\mathrm{Initialize}_{\theta,\phi}$, verifier V, compiler Γ; controller configuration $(\rho, \tau_{\mathrm{lat}}, T_{\max}, S_{\max})$. $S_{\max}$ is optional.

1:	Bind the task digest and run identities $\mathcal{R}$; initialize (A_x, H_0).	Initialize
2:	Decode and bind a_0 from $D_{\theta,\phi}(x, H_0)$; obtain $v_0 = V(x, a_0)$.	
3:	Form $B_0 = (a_0, H_0, v_0, \beta_0)$ with the canonical binding digest.	
4:	Set $E_0 = 0$, $s_0 = 0$, and $e_0 = \Gamma(v_0, v_0, 0, -1)$.	
5:	**if** $r_0 = 0$ **then return** $(B_0$, certified).	Check baseline
6:	**for** $t = 0, \ldots, T_{\max} - 1$ **do**	
7:	Validate the current run identities and retained bundle B_t.	Validate
8:	Obtain $\Delta_t = F_{\theta,\phi}(x, A_x, H_t, E_t, e_t, t)$ using private input copies.	Propose
9:	Check shape, dtype, device, finiteness, and $\max_\ell \|\Delta_{t,\ell}\|_2 \le \rho + \tau_{\mathrm{lat}}$.	Check legality
10:	Construct $H'_t = H_t + \Delta_t$; decode and bind a'_t from $D_{\theta,\phi}(x, H'_t)$.	Decode and bind
11:	Obtain $v'_t = V(x, a'_t)$; check result integrity and the fixed rank schema.	Verify
12:	Set $\chi_t = 1\{r'_t <_\rho r_t\}$.	Compare full ranks
13:	Compute $e_{t+1} = \Gamma(v_t, v'_t, \chi_t, t)$ before replacing the incumbent.	Compile feedback
14:	**if** $\chi_t = 1$ **then**	
15:	Form $B'_t = (a'_t, H'_t, v'_t, \beta'_t)$; set $B_{t+1} = B'_t$ atomically.	Commit
16:	Set $E_{t+1} = 0$ and $s_{t+1} = 0$.	
17:	**else**	
18:	Retain the exact bundle $B_{t+1} = B_t$.	Reject
19:	Set $E_{t+1} = \Delta_t$ and $s_{t+1} = s_t + 1$.	Exploration only
20:	**end if**	
21:	Validate the new state and record the attempted transition.	Audit
22:	**if** $r_{t+1} = 0$ **then return** $(B_{t+1}$, certified).	Zero rank first
23:	**if** $S_{\max}$ is enabled and $s_{t+1} \ge S_{\max}$ **then**	
24:	**return** $(B_{t+1}$, stalled_uncertified).	No completion claim
25:	**end if**	
26:	**end for**	
27:	**return** the retained bundle with status budget_exhausted_uncertified.	No completion claim

Notation: $B = (a, H, v, \beta)$ is the retained artifact–latent–verification bundle with binding digest; $r = v.\mathrm{rank}$. A_x is fixed; E_t stores only the most recent rejected displacement; e_t is the neural evidence feature. The strict protected order $<_p$ requires coordinate-wise non-regression and at least one strict improvement.

Execution contract: integrity checks surround interface calls. Contract violations raise errors rather than becoming ordinary rejection or success. Decoding and verification use fixed identities within a run. No controller-side projection is inserted. Report alignment: Algorithm 1 and Eqs. (1)–(16), (25).

3.10 Guarantees and their limits

Proposition 1. Protected non-regression. Under a fixed valid contract, every normal controller transition satisfies $r_{t+1} \preccurlyeq_P r_t$. If $\chi_t=1$, then $r_{t+1} \prec_P r_t$; if $\chi_t=0$, then $B_{t+1}=B_t$.

Proof. On acceptance, Equation (8) is true and Equation (15) installs the compared candidate bundle. Equation (7) therefore gives every required coordinate inequality and at least one strict inequality. On rejection, Equation (15) preserves the entire bundle and hence its rank. No property of the learned proposal distribution is needed. ▫

Proposition 2. Finite strict commitment. Let the initial rank have finite nonnegative integer coordinates. The number N_{commit} of strict commits in a run satisfies

$$\Phi(r_{t+1}) \le \Phi(r_t) - \chi_t, \qquad N_{\text{commit}} = \sum_t \chi_t \le \Phi(r_0). \tag{26}$$

Proof. A committed transition decreases at least one integer coordinate by at least one and increases none; a rejected transition leaves the potential unchanged. Summing the first inequality and using $\Phi \ge 0$ yields the bound. ▫

These are deterministic consequences of the transition rule for valid executions. They do not assume a contraction mapping, a successful optimizer, or a positive proposal success probability. The integer requirement matters: unrestricted nonnegative real coordinates could permit infinitely many strict decreases of vanishing size. The potential is a proof device; its decrease alone remains insufficient for commitment.

Finite commitment is not unconditional convergence to a solution. A decoder that maps every reachable latent to the same failing artifact produces no strict progress. Likewise, a task whose available repair path must first violate a protected obligation cannot be solved by committing that path under the fixed order. Rejected exploration can search for a direct admissible improvement, but a bounded proposal radius does not guarantee such a candidate exists. VGR deliberately exchanges unrestricted search trajectories for a protected retained state.

Proposition 3. Contract-relative completion. The controller returns certified completion exactly when the retained rank is zero. To infer a stronger semantic property $S(x,a)$, one additionally needs verifier soundness for that property:

$$r(x,a) = 0 \Rightarrow S(x,a) = 1. \tag{27}$$

Here $r(x,a)$ is the verifier's rank for task x and artifact a, and S is the intended Boolean correctness predicate. Passing a finite test suite certifies those test obligations. It is not automatically a proof of all possible program behaviors. The controller cannot supply missing verifier coverage.

Conditional reachability. An optional idealized analysis assumes unlimited attempts and no stall cutoff. Let $\mathscr{F}_n$ contain the complete history before an attempt at a nonzero incumbent. If there is a uniform $\varepsilon > 0$ such that

$$\Pr\left(r'_n <_P r_n \mid \mathscr{F}_n\right) \ge \varepsilon \tag{28}$$

for every reachable nonzero history, then the probability of h consecutive failures before the next commit is at most $(1-\varepsilon)^h$. A strict commit therefore occurs almost surely at each nonterminal stage, and Proposition 2 limits the number of such stages before zero rank. This establishes almost-sure certification in that idealized setting. It is a sufficient reachability assumption, not a property established by the proposal architecture or by a finite training run. Merely assigning positive probability to progress at every attempt is insufficient when those probabilities decay too rapidly. The implemented finite-budget controller makes no unconditional success claim.

3.11 Supervision interface and compatibility with post-training

Post-training operates on model parameters; VGR commitment operates on candidate states within a run. These are different update axes. A valid connection is to collect externally checked candidate transitions, freeze their labels under the protected order, and optimize a differentiable proposal objective. This process does not require gradients through the discrete verifier. It is applicable to a pretrained model whenever the required input serialization, target likelihood, and trainable parameter path exist.

Let c_t^{in} be the model-visible conditioning record built from the task, retained artifact, and permitted evidence. In a latent realization it also includes the accessible latent conditioning interface. A same-incumbent comparison pair consists of candidate sequences y^+, y^-, checked under the same task and verifier contract, with

$$r^+ <_P r_t, \qquad r^- \not<_P r_t. \tag{29}$$

The superscripts + and − identify accepted-progress and noncommittable examples, respectively. A rejected example can be unchanged, regressing, or incomparable. A valid improving candidate cannot be relabeled negative merely because another candidate is preferred. A final artifact

imported from an expert or another generator can be verified as supervision, but must not be represented as a latent transition emitted by the current proposer unless its latent-to-artifact origin is actually recorded.

For a candidate sequence of n target tokens, its teacher-forced log-probability is

$$\ell_\phi\left(y \mid c_t^{\text{in}}\right)=\sum_{k=1}^{n}\log p_{\theta_0,\phi}\left(y_k \mid c_t^{\text{in}}, y_{<k}\right). \tag{30}$$

θ_0 denotes fixed pretrained parameters, ϕ the trainable adaptation parameters, y_k the token at position k, and $y_{<k}$ its preceding target tokens. Prompt and diagnostic tokens are conditioning, not answer targets. The sequence sum above is distinct from an explicitly length-normalized variant; the actual training recipe must report which convention it uses.

The reference preference objective is

$$\begin{aligned} \ell^+ &= \ell_\phi\left(y^+ \mid c_t^{\text{in}}\right), \qquad \ell^- = \ell_\phi\left(y^- \mid c_t^{\text{in}}\right), \\ \mathcal{L}_{\text{pref}} &= \mathrm{E}_{\text{pairs}}\left[\text{softplus}\left(m_{\text{pref}}-(\ell^+-\ell^-)\right)\right]. \end{aligned} \tag{31}$$

The positive margin m_{pref} specifies the desired log-probability gap, $\text{softplus}(z)=\log(1+e^z)$, and the expectation is a minibatch average in the executable loss. This objective promotes relative preference for verified improvements. Alone, it does not guarantee a higher absolute probability for every accepted sequence or preservation of every pretrained capability.

Let D_{act} contain verified strict-progress actions and D_0 contain accepted zero-rank final artifacts. For each record, define a target-only negative log-likelihood

$$\mathrm{NLL}_\phi\left(y \mid c^{\text{in}}\right)=-\frac{\sum_{k=1}^{n} M_k \log p_{\theta_0,\phi}\left(y_k \mid c^{\text{in}}, y_{<k}\right)}{\sum_{k=1}^{n} M_k}, \qquad \sum_{k=1}^{n} M_k>0. \tag{32}$$

The mask $M_k\in\{0,1\}$ selects the supervised target tokens, and c^{in} is the corresponding model-visible conditioning record for that supervised example. The chosen reduction is made explicit here; the reference loss helper receives already computed action and generation losses and does not itself implement tokenization or masking. The two supervised terms are

$$\mathcal{L}_{\text{act}}=\mathrm{E}_{\mathrm{D}_{\text{act}}}\left[\mathrm{NLL}_\phi\right], \qquad \mathcal{L}_{\text{gen}}=\mathrm{E}_{\mathrm{D}_0}\left[\mathrm{NLL}_\phi\right], \tag{33}$$

$$\mathcal{L}=\lambda_p\mathcal{L}_{\text{pref}}+\lambda_a\mathcal{L}_{\text{act}}+\lambda_g\mathcal{L}_{\text{gen}}, \qquad \lambda_p,\lambda_a,\lambda_g\geq 0. \tag{34}$$

The coefficients weight preference, correction-action, and final-generation supervision. A nonzero-rank intermediate repair is valid action supervision but not a certified final-generation target. The recipe may activate a subset of these terms. Calling the pairwise helper requires a valid pair; a separate SFT-only path need not fabricate one. Dataset sizes, mixture weights, optimization schedules, and checkpoint selection are deferred to the post-training installment.

For a QLoRA-compatible weight matrix, one possible parameterization is

$$W_\phi=\text{Dequant}\left(Q_4(W_0)\right)+\frac{\alpha}{r_L}B_L A_L. \tag{35}$$

Here $W_0\in\mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ is a pretrained matrix; Q_4 is its frozen four-bit representation; Dequant denotes its computational reconstruction; $A_L\in\mathbb{R}^{r_L\times d_{\text{in}}}$ and $B_L\in\mathbb{R}^{d_{\text{out}}\times r_L}$ are trainable factors; r_L is the LoRA rank; and α is its scale [12, 13]. The ranks r_L and r_a need not match. This parameterization can train a token proposer or accompany a separately integrated latent operator. Quantization changes numerical computation and may change decoded proposals, but it does not change the protected comparison rule.

The runtime controller deliberately detaches protected tensors for authority separation. Consequently, calling the controller and backpropagating through its returned bundle is not a training implementation. Training must recompute differentiable likelihoods or proposal outputs through a dedicated path, treating verifier labels as fixed data. A claimed latent-operator experiment must additionally demonstrate a nonzero gradient path to that operator and deployment through the same decoder interface. Standard QLoRA on serialized verified examples establishes adaptation of the token model, not that additional architectural claim.

3.11.1 Why VGR is naturally a post-training mechanism

VGR is most naturally applied after foundation pretraining because its supervision signal is defined over task-level state transitions, not over isolated next-token events. A VGR training record requires a model that can already materialize a candidate artifact, a stable task identity, an immutable verifier contract, an incumbent state against which the candidate is compared, and a receipt that distinguishes strict progress, non-progress, regression, incomparability, and zero-rank completion. These objects are available at the granularity of downstream trajectories; they are generally absent from web-scale next-token pretraining. Treating task-specific verifiers as a foundation-pretraining objective would also make the learning loop depend on expensive executable checks whose semantics differ across code, mathematics, tools, and scientific domains.

Continued pretraining or domain mid-training can increase exposure to specialized distributions, but exposure alone does not create the same-incumbent counterfactual labels required by the protected order. The distinctive VGR signal is relational: candidate y+ is preferred because the external contract establishes r+ $<$ P r for the same incumbent, whereas y− is noncommittable because it is unchanged, regressing, or incomparable. Verified strict-progress actions can therefore supervise correction behavior, zero-rank artifacts can supervise final generation, and rejected candidates remain useful negatives without becoming retained state. This is precisely the regime in which post-training can reshape the proposal distribution while leaving the discrete verifier and commit rule outside the gradient path.

The resulting division of labor is structural rather than merely procedural. Pretraining supplies broad linguistic, code, mathematical, and scientific competence; VGR-guided post-training increases the probability that a capable model proposes transitions compatible with externally checkable progress; deployment-time governance preserves the hard acceptance invariant. Moving the evidence logic earlier into generic pretraining would either discard the task-specific semantics that make the labels meaningful or impose verifier execution at foundation-model scale. Moving it entirely to prompt-time orchestration would preserve governance but forgo internalization of the correction, diagnosis, and proposal patterns exposed by verified trajectories. VGR therefore occupies the post-training interface at which learned proposal quality and external persistence authority can be improved jointly without being conflated.

3.12 Integration contract and present claim boundary

An executable Qwen latent realization must identify the controlled activation location, the source-anchor construction, the latent shape, and how the decoder consumes the changed activation. All downstream cache entries dependent on that activation must be recomputed or restored consistently. Otherwise, a stale KV cache could make the produced artifact correspond to a different computational state from the one recorded in the bundle. A hidden-state hook alone does not establish this end-to-end consistency.

Each rollout uses a fixed model snapshot. For VeriLoop E2 runs, the model-contract identity binds the post-trained Qwen 3.8-27B checkpoint used to generate proposals and model-side reasoning traces; changing back to the untouched base checkpoint creates a different model identity and therefore a different run provenance. Updating adapters between training steps likewise creates a new model identity for subsequent rollouts and must not silently change the decoder inside an existing certified state chain. Similarly, a new test suite or changed obligation meaning creates a new verification contract and requires re-evaluation rather than comparison with stale ranks. These identity boundaries are what make capability attribution auditable: they determine which checkpoint produced a proposal, while leaving acceptance authority with the external verifier.

The reference implementation provides the abstract model interfaces, protected controller, low-rank proposal module, evidence compilers, and a loss helper. Instantiating the latent VGR interface in Qwen additionally requires a concrete model adapter, a consistent cache implementation, and execution traces showing that the latent module participates in training and inference. The abstract interfaces alone do not establish those integration facts. Accordingly, this installment separates two claims that must not be conflated. First, VeriLoop E2 itself is the post-

trained Qwen 3.8-27B checkpoint used as the model-side reasoning engine in the reported E2 research trajectories. Second, the full latent VGR realization is an additional architectural claim that requires its own integration evidence. The scientific reasoning demos in Section 5 establish the former through bound model identity and recorded E2/Harness trajectories; they do not, by themselves, establish that the latent proposal operator of Section 3.8 was active inside the backbone during those derivations.

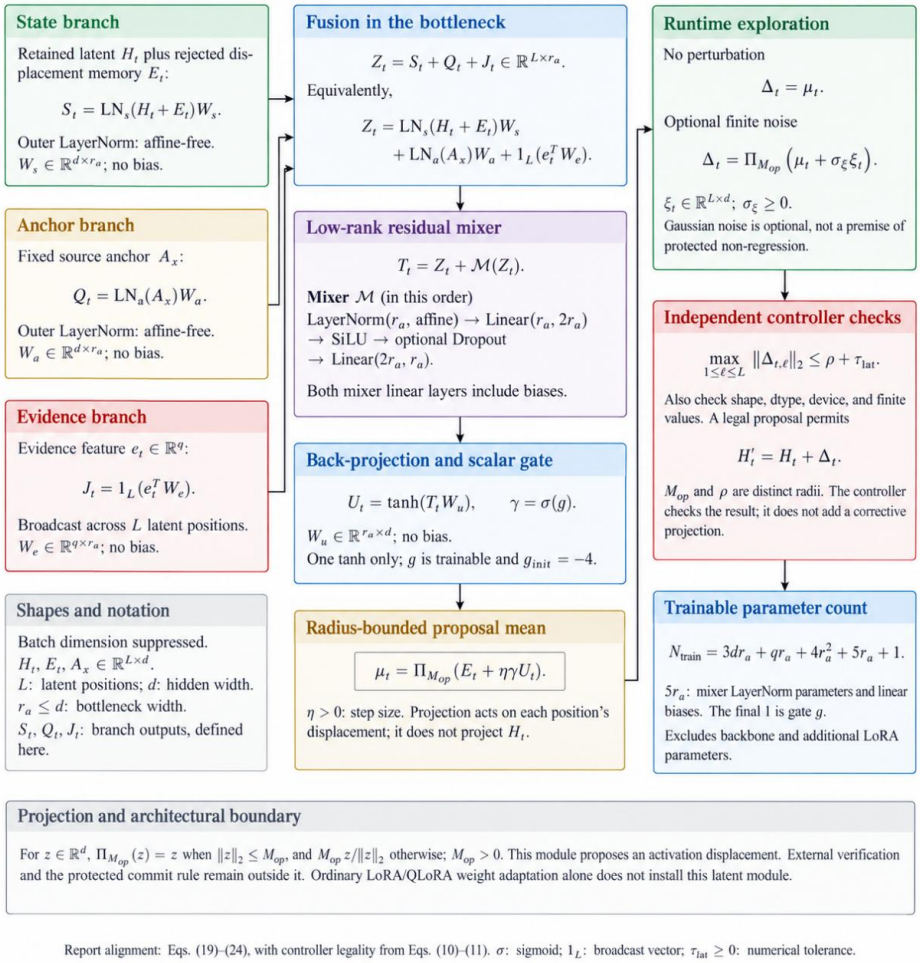


Figure 8 | Reference low-rank latent proposal operator. State, anchor, and evidence branches are fused through the bottleneck mixer, back-projected, gated, and projected to the internal radius before optional exploration noise. The external controller independently enforces proposal legality; the internal projection radius and controller legality radius remain distinct.

Within that boundary, the attribution rule is precise. Post-training changes the model parameters and therefore the distribution of proposals, abstractions, diagnoses, and next-step decisions produced by the E2 checkpoint; the deterministic Harness does not become part of those learned parameters. Conversely, the Harness supplies execution, exact or interval verification, rollback, certificate replay, and persistence control, so an evidence-closed scientific result is a system result rather than a model-only proof. What can be attributed directly to the post-trained E2 checkpoint is the model-side reasoning trajectory generated under its bound model identity. What can be attributed directly to the Harness is the enforcement and certification layer. What remains an empirical causal question is how much better the post-trained checkpoint performs than the raw Qwen 3.8-27B base under matched conditions. This distinction is the basis for interpreting both the benchmark campaign and the two demos without collapsing model capability, system governance, and causal training gain into one claim. Within this report, that division between learned reasoning and external persistence authority is the concrete systems path toward verifiable intelligence.

5 Evaluation and Scientific Reasoning Demos

Sections 5.1 and 5.2 report the finalized benchmark campaign and the frozen post-training evidence package. Every benchmark result is tied to the evaluated post-trained VeriLoop E2 checkpoint and to the evaluation contract used on that track; the untouched Qwen 3.8-27B base remains a separate baseline identity and is never relabeled as E2. Checkpoint-only evaluation is model-level evidence. Where Harness-side execution or verification participates, the result is evidence for that evaluated E2 system configuration and the Harness contribution remains explicit. A claim that post-training itself caused a benchmark improvement requires a matched raw-base-versus-E2 experiment under the same contract. The post-training measurements below therefore establish optimization dynamics, data provenance, recorded recipe evidence, and model lineage; they are not converted into an unsupported causal benchmark delta.

The two demos use the trajectory/system realization of VGR acceptance semantics. In both, the VeriLoop E2 checkpoint is the model-side actor that proposes mathematical or physical constructions, abstracts intermediate structure, diagnoses failed routes, and selects the next reasoning direction. The team's internal trial Harness supplies the complementary authority: it executes external checks, organizes exact and interval certificates, records counterexamples, enforces rollback, and freezes only states whose declared obligations close. This establishes a clean provenance chain—E2 for model-side reasoning behavior, Harness for verification and persistence—without turning the Harness into the model or treating model confidence as a certificate.

5.1 Benchmark Evaluation

5.1.1 Evaluation contract and score construction

The final evaluation matrix contains nine benchmark tracks spanning repository-level software repair, terminal/agentic execution, mathematical reasoning, scientific question answering, and competition mathematics. The matrix is intentionally heterogeneous: each benchmark retains its native unit of evaluation and native pass criterion. We therefore report each denominator and execution protocol explicitly and do not collapse the nine tracks into a macro-average, which would assign arbitrary weights to incomparable scoring contracts.

The public disclosure layer is receipt-oriented rather than answer-oriented. Task packages bind the submitted artifact and official evaluation receipt by SHA-256 while omitting answer payloads, patches, prompts, raw responses, reasoning traces, and raw evaluator logs. For SWE-bench Pro, DeepSWE v1.1, and Terminal-Bench 2.1, the frozen aggregate is computed over the complete task set. For Terminal-Bench 3.0 and 4.0, public per-task directories may expose only verified passing-instance evidence; the aggregate scores below therefore come from the frozen full-campaign result rather than from counting the passed-instance disclosure subset. This distinction prevents a disclosure view from being mistaken for the benchmark denominator.

Table 5.1 | Final VeriLoop E2 benchmark campaign. The denominator is preserved in the benchmark-native unit; multi-run mathematics tracks count every recorded run rather than selecting a best run.

Benchmark	Evaluation protocol	Passed / total	Score
SWE-bench Pro	731 tasks; 1 run/task	557 / 731	76.20%
DeepSWE v1.1	113 tasks; 1 run/task	73 / 113	64.60%
Terminal-Bench 2.1	89 tasks; 1 run/task	79 / 89	88.76%
Terminal-Bench 3.0	74 tasks; 1 run/task	22 / 74	29.73%
Terminal-Bench 4.0	66 tasks; 1 run/task	25 / 66	37.88%
AIME 2026	30 problems × 16 runs	472 / 480	98.33%
ArXivMath	103 problems × 4 runs	412 / 412	100.00%
GPQA Diamond	198 questions; 1 run/question	186 / 198	93.94%

Benchmark	Evaluation protocol	Passed / total	Score
Apex 2025	12 problems × 4 runs	43 / 48	89.58%

For GPQA Diamond, the report uses the frozen post-adjudication result of 186/198 = 93.939394%. Twelve items—040, 043, 045, 050, 053, 059, 066, 068, 069, 078, 082, and 084—were corrected by the final adjudication pass; the older task-level evaluator status is therefore not re-summed. The frozen run records model identity veriloop-e2, Inspect AI 0.3.262, the official source commit 56686c06f5e19865c153de0fdb11be3890014df7, temperature 0.6, seed 0, a 4096-token generation cap, and a 2400 s task timeout. The public release package is content-addressed by SHA-256 62a4d197e72c6e7106cabf4f43b3906a924b95ef9ef5f2e234e1f661c2b1d3b5.

5.1.2 Code repair and agentic execution

The code and agentic tracks probe different failure surfaces. SWE-bench Pro measures repository-level repair under an official patch-evaluation contract and yields 557/731 = 76.20%. DeepSWE v1.1 yields 73/113 = 64.60%, providing a second software-engineering measure with a distinct task distribution and evaluation surface. Terminal-Bench 2.1 yields 79/89 = 88.76%, while the materially different Terminal-Bench 3.0 and 4.0 versions yield 22/74 = 29.73% and 25/66 = 37.88%, respectively. These versioned Terminal-Bench results are reported independently rather than interpreted as a monotone time series: task inventories and solution contracts change across releases, so score differences across versions do not isolate a single capability delta.

The code-side results are relevant to the VGR design because they require more than endpoint text matching. Repository repair and terminal tasks expose executable consequences, making regression, incomplete fixes, tool misuse, and malformed artifacts observable under external checks. At the same time, the benchmark table is an evaluation result rather than a proof that the VGR acceptance rule caused those scores. The attribution is deliberately narrower: these are outcomes of the post-trained E2 checkpoint under the recorded evaluation configuration; causal gain from post-training remains a separate matched-comparison question.

5.1.3 Mathematical and scientific reasoning

The mathematics/science block was evaluated with protocols designed to avoid best-of-run inflation. AIME 2026 contains 30 problems, each executed for 16 recorded runs; all 480 scored units are retained, of which 472 pass, for 98.33%. ArXivMath contains 103 problems with four recorded runs each; all 412 units pass. Apex 2025 contains 12 problems with four runs each and closes at 43/48 = 89.58%. GPQA Diamond is evaluated once per question and, after the frozen adjudication described above, closes at 186/198 = 93.94%. The multi-run figures therefore measure repeated execution under a fixed protocol rather than a selectively reported maximum.

The four tracks complement the code benchmarks by changing the verifier surface. AIME and Apex emphasize competition-style derivation under repeated sampling, ArXivMath stresses mathematical problem solving over a larger problem set, and GPQA Diamond tests high-difficulty scientific knowledge and reasoning. Their role in this report is breadth, not aggregation: the central question is whether the same post-trained checkpoint remains effective when the externally checkable notion of success changes substantially across domains.

5.1.4 Cross-model public-score context

To place the absolute VeriLoop E2 results in external context, Figure 1 provides a benchmark-matched comparison against selected public frontier-model results on evaluation families for which a directly reported public score is available in the comparison set. The comparison is supplementary to Table 5.1 rather than a redefinition of the finalized nine-track E2 campaign: public competitor coverage is not uniform across benchmarks, and the panel set therefore need not coincide one-for-one with the internal campaign matrix. Comparisons are interpreted only within

the same named benchmark and version. We do not construct a cross-benchmark rank, normalized aggregate, or global model score from these heterogeneous panels.

The comparison is also descriptive rather than causal. Public results may differ in tool access, execution environment, evaluator implementation, sampling policy, and reporting date unless those conditions are explicitly aligned by the source. Figure 1 therefore answers a limited question—where the frozen E2 score lies relative to selected published results on the same benchmark—not whether E2 is globally superior to another model and not how much of the E2 result was caused by post-training. Table 5.1 remains authoritative for VeriLoop E2's own frozen scores, denominators, and execution protocols.

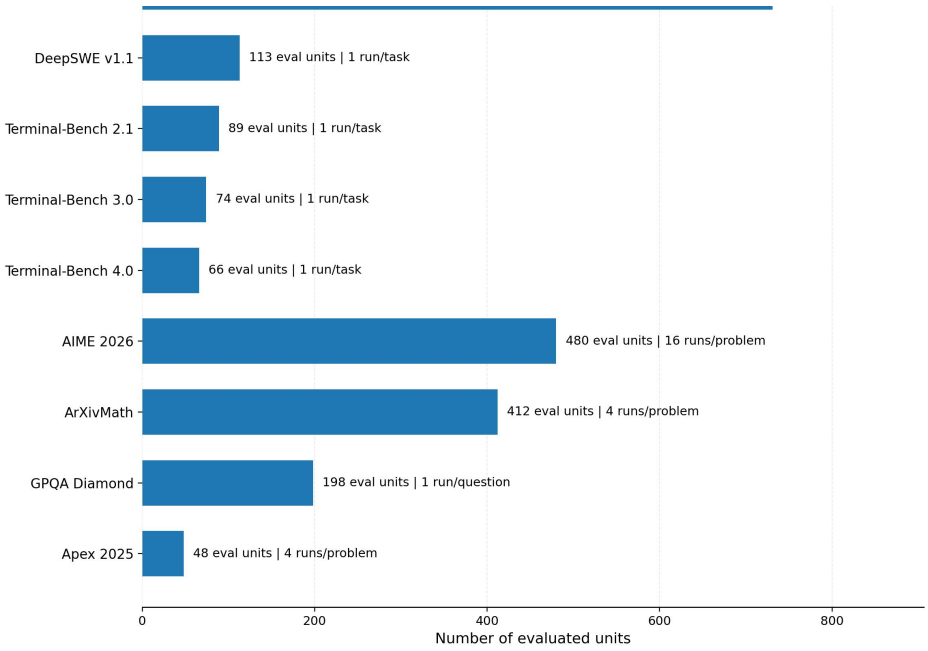


Figure 9 | Evaluation coverage and execution protocol. The figure should display the full benchmark denominators: 731, 113, 89, 74, 66, 480, 412, 198, and 48 scored units for SWE-bench Pro, DeepSWE v1.1, Terminal-Bench 2.1/3.0/4.0, AIME 2026, ArXivMath, GPQA Diamond, and Apex 2025, respectively.

5.1.5 Interpretation and attribution boundary

The benchmark campaign supports a checkpoint-level capability claim and, where external execution is part of the benchmark path, a system-configuration claim: the evaluated model actor is the post-trained VeriLoop E2 checkpoint, while benchmark-native tools and the internal Harness retain their execution and verification roles. The campaign does not establish that the raw Qwen 3.8-27B checkpoint would obtain lower scores under an otherwise identical protocol, because the frozen evidence package does not contain a matched same-contract base-versus-E2 campaign. We therefore do not report an artificial “post-training gain” column. This constraint is important: provenance answers which model produced the evaluated behavior; only a controlled counterfactual answers how much of the result was caused by post-training.

5.2 Post-Training Optimization and Evidence

5.2.1 Evidence base, data composition, and reporting policy

The frozen post-training evidence package records 1,841,831 unique source records across 434 unique source files. Because a source may be routed into more than one capability surface, the corresponding cross-surface reuse mass is 5,731,380 records; this larger quantity is not a second unique-data total. The archive uses “record” as the source-example unit and does not equate

records with tokens. The six capability surfaces analyzed below share this source mixture through controlled routing and replay.

The recorded source mixture spans repository repair and tool use (including Nemotron-SFT-SWE-v3, SWE-Hero/OpenHands trajectories, SWE-bench Verified, Aider Polyglot, BigCodeBench, and Terminal-Bench 2.0), mathematical reasoning (OpenR1-Math-220k and OpenMathReasoning), scientific reasoning (Nemotron science/physics, VERIQ, and verifiable QFT), and project-specific verifier-governed trajectories. Surface-specific routing weights emphasize different subsets while protected-core replay remains part of the training contract. The resulting corpus is therefore a structured multi-surface mixture rather than a collection in which every record belongs uniquely to one capability surface.

The reporting policy is evidence-first. Recipe fields are stated only when an authoritative artifact records them: QLoRA-4 quantization, bfloat16 computation, gradient checkpointing, per-device batch size 1, gradient accumulation 4, seed 20260818, and surface-specific learning rates are recorded. LoRA rank, LoRA alpha, optimizer, sequence length, and total post-training step count are not authoritatively present in the frozen package and are therefore left NOT_RECORDED rather than reconstructed from defaults or recollection. Missing metadata is treated as an evidence boundary rather than inferred after the fact.

5.2.2 General-capability optimization dynamics and evidence interpretation

Table 5.2 and Figure 10 summarize the six capability surfaces used in VeriLoop E2 post-training. Each surface contains twelve recorded evaluation-loss observations. Across all six surfaces, the final recorded evaluation loss is lower than the first: Uncertainty decreases from 0.9125 to 0.2763 (−69.72%), Evidence Binding from 0.7600 to 0.2435 (−67.96%), Rollback from 0.8023 to 0.3362 (−58.10%), ToolSpec from 0.4071 to 0.1810 (−55.54%), VGR recurrence from 0.4324 to 0.2226 (−48.52%), and Domain Core from 0.8544 to 0.6078 (−28.86%). Figure 10 makes the common first-to-last optimization trend visible while preserving the substantial differences in initial loss and reduction magnitude across capability surfaces. These measurements characterize optimization dynamics within the recorded post-training process; they are neither downstream benchmark-accuracy improvements nor a causal estimate of gain over the untouched Qwen 3.8-27B base checkpoint.

Table 5.2 | General-capability post-training evaluation-loss dynamics.

Capability surface	First eval loss	Last eval loss	Minimum	Relative change
VGR recurrence	0.4324	0.2226	0.2186	−48.52%
Domain Core	0.8544	0.6078	0.6078	−28.86%
Uncertainty	0.9125	0.2763	0.2763	−69.72%
Evidence Binding	0.7600	0.2435	0.2434	−67.96%
Rollback	0.8023	0.3362	0.3362	−58.10%
ToolSpec	0.4071	0.1810	0.1810	−55.54%

The final value should not be confused with the minimum observed checkpoint. VGR recurrence reaches a minimum of 0.2186 before ending at 0.2226, and Evidence Binding reaches 0.2434 before ending at 0.2435; the other four surfaces end at their recorded minima. Accordingly, the evidence supports a strong net optimization trend from the first to the last recorded evaluation point, but it does not justify the stronger claim that every intermediate evaluation decreases monotonically. This distinction is retained because checkpoint-level variation is part of the measured training dynamics rather than noise to be edited out of the report.

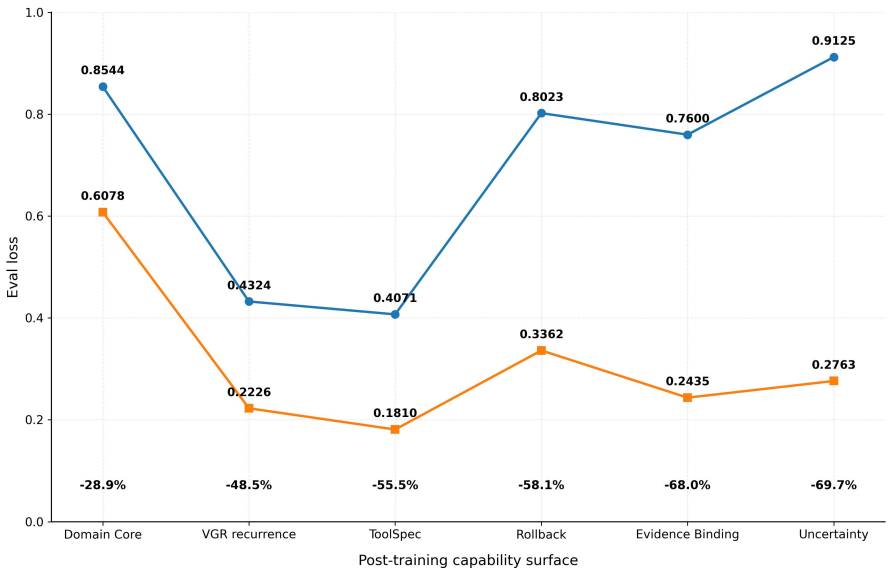


Figure 10 | General-Capability Post-Training Optimization Dynamics. First and last recorded evaluation loss across the six general-capability surfaces in the recorded post-training process. All six surfaces show lower final than initial evaluation loss; percentages denote within-surface first-to-last reductions and should not be interpreted as downstream benchmark gains.

The recorded learning rates are surface specific rather than normalized into a single recipe: 1.667×10^{-7} for VGR recurrence, 1.333×10^{-6} for Domain Core, and 2.667×10^{-7} for Uncertainty, Evidence Binding, Rollback, and ToolSpec at the cited recorded points. This heterogeneity is consistent with the training design: individual capability surfaces may optimize under different local schedules while the evidence contract requires that each reported value remain tied to its recorded source rather than to a reconstructed global recipe.

Table 5.3 | Recorded post-training recipe fields and evidence status. Fields absent from authoritative artifacts remain explicitly unreported.

Field	Recorded value	Status
Quantization / compute	QLoRA-4 / bfloat16	RECORDED
Memory optimization	Gradient checkpointing enabled	RECORDED
Batching	Per-device batch 1; accumulation 4	RECORDED
Seed	20260818	RECORDED
Learning rate	Surface specific; values reported above	RECORDED
Undocumented recipe fields	LoRA rank/alpha; optimizer; sequence length; total post-training steps	NOT_RECORDED

Taken together, the post-training evidence establishes three points. First, all six general-capability surfaces show substantial first-to-last evaluation-loss reduction. Second, the data and recipe ledger distinguishes unique-source mass from cross-surface reuse and distinguishes recorded hyperparameters from undocumented ones. Third, these measurements characterize optimization behavior and provenance rather than causal benchmark uplift. The frozen causal-status record does not establish a matched raw-base-versus-E2 comparison; therefore no artificial "post-training gain" is derived from these curves. A causal delta requires rerunning the untouched Qwen 3.8-27B checkpoint under the same benchmark, decoding, Harness, and evaluator contracts.

5.3 Demo I: Riemann ζ Zero Proportion — From a Public Baseline to a Strict 67.35% Finite-Dimensional Certificate

5.3.1 Demo setup: research contract, model role, and protected quantity

The first demo uses the VeriLoop E2 checkpoint to continue from the public Zeta23 line rather than reproduce it. At the time of the campaign, the external kernel-verified ledger value was 67.2500703679%, obtained in the Anthropic 2026 Zeta23 program and tracked by the corresponding challenge infrastructure. E2 was given that public baseline, the fixed analytic interface, and the requirement that every claimed improvement survive independent certificate checks orchestrated by the team's internal trial Harness. The model's task was not to guess a larger percentage, but to reason over the proof structure, propose finite-dimensional witnesses, diagnose failed routes, and redirect the search when the verifier showed that an apparent gain could not be certified. The retained objective was the proportion of nontrivial zeros that are both simple and on the critical line,

$$\kappa=\liminf_{T\to\infty}\frac{S(T)}{N(T)},$$

$N(T)0<\mathrm{Im}\rho\le TS(T)\mathrm{Re}(s)=1/2$where the numerator counts nontrivial zeros that are simple and lie on the critical line and the denominator counts the relevant nontrivial zeros in the same asymptotic regime. The quantity is a lower bound on a zero proportion; it is not a percentage of the Riemann Hypothesis proved and does not assert the Riemann Hypothesis. This scope condition was itself protected throughout the demo, so a numerically stronger candidate could not be retained by weakening the statement being certified.

The upstream analytic interface retained by the campaign has the form

$$S\ge H(v)N+\Delta(M)-o(N),\qquad H(v)=2-\frac{I_2+J}{I_1^2},$$

$v[-1/2,1/2]M\Delta(M)\Delta(M)$ with an even nonnegative window supported on the prescribed interval and a positive-semidefinite matrix supplied by the pair-correlation/explicit-formula layer. The external baseline is recovered by discarding the nonnegative matrix correction. This immediately gave E2 a concrete reasoning target: the window-only route was already represented by the public baseline, so a meaningful advance had to come from a new, independently checkable contribution rather than from relabeling the inherited term.

$H(v)$ That observation became the first protected obligation of the demo. E2 was free to explore alternative witnesses and decompositions, but the Harness would retain a candidate only if the analytic interface, finite-dimensional identities, local inequalities, exact assembly, and scope boundary remained mutually consistent. The model therefore reasoned inside a fixed acceptance contract: exploration was open-ended, while promotion of a research state was not.

5.3.2 Finite-dimensional reduction: how E2 changed the search target

The retained finite-dimensional chain begins with the spectral functional

$$\Psi(t)=\begin{cases}(t-1)^2, & 0\le t\le 2,\\ 2t-3, & t\ge 2,\end{cases}\qquad \Delta(G)=\mathrm{tr}\Psi(G).$$

For $X = G - I$, and with U denoting the positive spectral part of $G - 2I$, the campaign used the proved identity

$$\Delta(G)=//X-U//_F^2+2\mathrm{tr}U.$$

A retained bandwidth q induces a graph of maximum degree $d=2q$ and chromatic number $r=q+1$. With $T=r/(r-1)=(q+1)/q$ and retained energy

$$E=2\sum_{1\le j-i\le q}\left|G_{ij}\right|^{2},$$

the matrix theorem proved in the archive gives

$$\Delta(G)\ge \mathrm{h}(E),$$

where

$$\mathrm{h}(E)=\begin{cases}E, & 0\le E\le T,\\ E-\dfrac{d}{d+1}(\sqrt{E}-\sqrt{T})^{2}, & E\ge T.\end{cases}$$

$T\mathrm{h}(E)=EE>T$ After the matrix reduction was established, E2 made a consequential negative decision. The binding witness lay below the branch point at which the alternative part of the bound could help, so further effort spent sharpening that branch could not improve the protected objective. The model therefore redirected the search toward the matrix/pressure contribution that still had headroom. This was not a heuristic preference imposed after the fact: the verifier exposed that one direction was mathematically non-binding, and E2 changed the search target accordingly.

The next proposal was to express the remaining local obligation through pressure weights b_r, pair weights a_{ij} satisfying the span-capacity identity, and the kernel $W(x) = (K(x)/K(0))^2$. For a gap vector g in the nonnegative orthant $[0,\infty)^q$ and slope $s > 0$, each candidate certificate had to satisfy the following inequality under the same Harness-controlled contract:

$$P_{\mathrm{loc}}(g)+sQ_{\mathrm{loc}}(g)\ge\varepsilon_s \qquad \text{for all} g\in[0,\infty)^q.$$

After translating these local inequalities across a block, the certificates yield

$$p(E)=\max\left(0,\max_{s}\{n\varepsilon_s-sE\}\right), \qquad n=m-q,$$

and the finite assembly reduces to

$$R=\inf_{E\ge0}[\mathrm{h}(E)+\eta p(E)].$$

Because the infimum is attained on a finite kink set, R can be evaluated exactly. The resulting lower bound is

$$\kappa\ge C:=\frac{mH-\eta B(m-q)}{m-R},$$

$B=\sum_r b_r$ This reduction created the narrow interface that governed the remainder of the demo. Once the upstream analytic layer was fixed, every E2-generated witness—regardless of how it was discovered—had to reduce to certified local inequalities and exact rational assembly. That separation was crucial: E2 could search aggressively in the proposal space, but the internal Harness could freeze a candidate only after the same finite-dimensional obligations closed.

5.3.3 Counterexample-driven rejection: why two stronger-looking candidates were discarded

$\{1.04,1.975,2.915\}^q 5.1\times10^{-5}$ E2 initially produced two numerically attractive 67.35%-class candidates, 67.350352375073% and 67.35006335392536%. Neither was retained. When the Harness expanded the multistart search and rechecked the local minima, both candidates were refuted. Their failure exposed a structural bias in the proposal distribution: the search family overrepresented alternating gaps near 1.04 and 1.975 and missed minimizers containing a third characteristic gap near 2.91. Once the seed family was expanded to the complete ternary structure, several estimated minima fell by far more than the candidates' margins.

$q1.2\times10^{-5}$ This is where the demo becomes a test of model reasoning rather than decimal optimization. E2 did not patch the displayed percentage or search for a narrative that preserved it. The failed candidates were rolled back, the missing 2.91-scale geometry was promoted into the next search state, and the model regenerated its proposals under the stronger condition. A second false-progress source—a warm evaluator with measurable noise—was likewise removed from the acceptance path. Counterexamples therefore changed what E2 searched for next, while the last verified state remained intact.

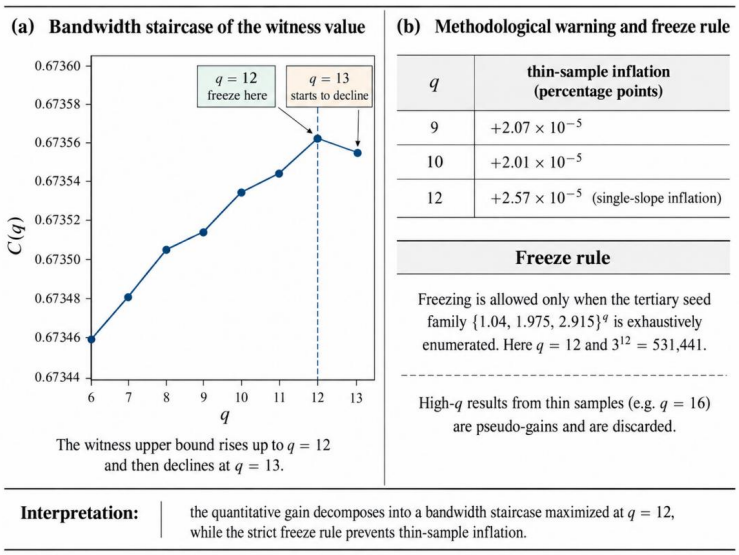


Figure 11 | Search-stage witness bandwidth staircase and evidence-governed freeze rule. The witness value increases with q up to the exploratory q=12 configuration, while thin-sample inflation and incomplete seed coverage are explicitly excluded from the acceptance path. This figure describes the search and freeze diagnostics rather than the final strict certificate, which is established separately at q=8.

The same model–Harness loop exposed three bottlenecks inside the interval verifier. E2 first diagnosed whether each apparent stall was mathematical or merely numerical/representational; the Harness then tested the proposed repair under fail-closed replay:

Reading a cell minimum from a first-order table incurred an error of order $h|W'|\approx2.4\times10^{-4}$, already larger than the target margin. More nodes could not repair that loss. The accepted replacement used second-order interpolation with a rigorous remainder, reducing this error to roughly 1.5×10^{-7} at comparable table size.

A tangent-plane test gated on a positive-definite lower Hessian almost never fired. It was replaced by an always-valid αBB lower bound with the smallest eigenvalue bounded by Gershgorin disks.

Single-precision box endpoints introduced an approximately 10^{-6} width floor near coordinate values around 10. Boxes that appeared mathematically unresolved were in fact storage-limited. The verifier was moved to double-precision endpoint storage.

The first strict closure in this lineage was therefore achieved by changing the reason the verifier stalled, not by blindly increasing compute. E2 identified that more nodes could not overcome the first-order interpolation error floor, that the positive-definiteness gate was formally valid but operationally inert, and that single-precision endpoints created a storage floor. After those failure modes were isolated, the accepted verifier path used second-order interpolation with a rigorous remainder, an always-valid αBB lower bound, and double-precision endpoint storage.

With the verifier repaired, the first local inequality to close was the $s = 1/2$ obligation at $\varepsilon = 0.0067328$. The checker returned PROVED after 43,685,299 nodes in 198 s. E2 treated this as a genuine state improvement but not as a final result: one protected coordinate had closed while the

remaining obligations were still open. The internal Harness therefore retained the certificate as reusable evidence and kept the 67.35%-class claim unpromoted.

5.3.4 Closed checkpoints and proof-cost-aware retention

The campaign then accumulated several evidence states rather than one monotone sequence of increasingly large decimals. This distinction matters for the demo because E2 was asked to reason about both mathematical value and proof status. A partial lock, a proof-design optimum, a historical closure, a low-cost replay tier, and a strict re-certification are different states even when some share the same or a lower numerical value.

Table 5.4 | Evidence-state lineage of the Riemann ζ campaign. Numerical value and evidence status are separated so that partial closure, proof-design closure, historical certification, proof-cost optimization, and strict re-certification are not conflated.

Evidence state	Value	Evidence status	Interpretation
Earlier q=6 predecessor	67.34574870891890%	Earlier exhaustive interval certificates completed; raw artifacts are not re-established in the present release	Historical closed checkpoint carried forward by provenance
q=8 partial lock	67.350000084257616%	Exact assembly frozen; s=1/2 certificate PROVED; 3 of 4 then-required local obligations still open	A genuine evidence advance, but explicitly not a completed finite-dimensional certificate
q=8 proof-optimal four-slope design	67.350000755825732%	Exact-rational slope-set/margin optimization; maximum uniform margin $\approx 3.084\times10^{-5}$ for {1/2,19/20,1,21/20}	Closed design calculation used to choose the proof geometry; not promoted as a theorem checkpoint
Historical q=8 bb9 closure	67.350003708785593%	3 local inequalities; 48/48 shards PROVED; 428,006,362 nodes; exact rational assembly	Finite-dimensional local layer closed under the historical bb9 acceptance path
Cost-optimized q=8 short tier	67.275055959117140%	4/4 inequalities freshly replayed; exact rational assembly; 793,374 total nodes	Lower numerical value, but a self-contained low-cost certificate above the then-current 67.2500703679% external threshold
Strict q=8 re-certification	67.350003708785593%	3/3 strict local inequalities PROVED; 327/327 well boxes PROVED; 190,375,830 total strict B&B nodes; exact assembly PASS	Primary high-value finite-dimensional result of the frozen strict release; same decimal, stronger trust boundary

The q=8 route therefore did not jump directly from a candidate decimal to a completed proof. The partial lock at 67.350000084257616% recorded one local certificate closed while three remained open. E2 then used exact-rational margin analysis to redesign the slope set rather than merely increase search depth. The four-slope set {1/2, 19/20, 1, 21/20} preserved the 67.35% threshold with a maximum uniform margin of approximately 3.084×10^{-5}, yielding 67.350000755825732% as a proof-design result. The Harness kept that state explicitly labeled as design evidence rather than promoting it to a completed theorem.

The same analysis revealed a proof-cost asymmetry. Removing s = 21/20 sacrificed only about 1.5×10^{-7} of available margin while eliminating an entire local certificate. E2 therefore proposed a three-slope q=8 route that preserved the target while reducing the verification burden. The already-proved s = 1/2 inequality was reused as protected evidence instead of being recomputed for presentation. This is the practical recurrence pattern of the demo: a verified component survives into the next state, while only the unresolved part is redesigned.

$\varepsilon_s s = 12.1\times10^{8}\varepsilon = 0.0080132\varepsilon = 0.0053$ The short tier made that reasoning policy even clearer. E2 recognized that the external comparison threshold did not require defending every decimal of the high-value witness, so it solved backward from the threshold to the loosest certificate parameters that still preserved a strict exact-rational improvement. The resulting 67.275055959117140% tier closed with only 793,374 local branch-and-bound nodes. It was retained not as the numerical frontier, but as a separate low-cost reproducibility tier. The lesson extracted by the model was margin-driven certification: prove only as tightly as the actual claim requires.

Table 5.5 | Cost-optimized q=8 short-tier replay. All four local inequalities terminate fail-closed with stack_left=0 and HARD=0; their node counts sum to 793,374.

Slope s	ε	Nodes	Terminal state
1/2	279/50000 = 0.00558	12,270	PROVED; stack_left=0; HARD=0
19/20	33669/5000000 = 0.0067338	212,210	PROVED; stack_left=0; HARD=0
1	34219/5000000 = 0.0068438	265,637	PROVED; stack_left=0; HARD=0
21/20	17333/2500000 = 0.0069332	303,257	PROVED; stack_left=0; HARD=0
Total		793,374	4/4 PROVED

For the E2 demo, this separation is more informative than a single best number. The protected research state contains numerical value, proof status, trust boundary, and proof cost. A lower numerical candidate can be worth retaining as a reproducibility tier when it strictly improves verification cost without being mislabeled as the frontier; a higher candidate with an unresolved certificate remains exploratory. The Harness enforces those labels, while E2 uses them to decide what kind of proposal should come next.

5.3.5 Strict closure: how E2 and the Harness froze 67.350003708785593%

The frozen strict release replaces the historical acceptance path with a smaller fail-closed trust boundary. The witness uses q=8, m=531, η=1, and the exact final fraction

$$C=\frac{722547711262091300265625000000000}{10728250504429256670617146151736 41}=67.350003708785593\%.$$

Its three local inequalities close as follows.

Table 5.6 | Strict q=8 local-certificate closure for the frozen 67.350003708785593% witness.

Slope s	ε_s	Strict nodes	Acceptance path	Terminal state
1/2	526/78125	23,644,472	Direct strict interval branch-and-bound; no well shortcut	stack_left=0; HARD=0; PROVED
19/20	9879/1250000	75,294,070	Independently certified wells plus global complement	stack_left=0; HARD=0; PROVED
1	20033/2500000	91,437,288	Independently certified wells plus global complement; 5 exact root shards	5/5 shards PROVED; aggregate PROVED
Total		190,375,830		3/3 PROVED

The numerical witness had already closed under the historical bb9 path, where three eight-dimensional inequalities were split into 48 shards and all 48 returned PROVED. E2 did not treat that as the end of the demo. The next reasoning step was to ask whether the same value could survive a smaller and more explicit trust boundary. The strict route therefore replaced the historical acceptance path with outward-rounded storage, interval transcendental evaluation, independently certified difficult wells, exact symmetry preflights, and fail-closed global-complement checks. The same witness then re-closed with 190,375,830 strict branch-and-bound nodes and 327/327 difficult wells proved. The value was unchanged; the evidence state was stronger.

πRCC>0.6735Before the Harness allows a global run to prune a box through a local-well shortcut, a separate checker must certify that well from exact-rational configuration data. The frozen batch is 327/327 PROVED with zero failures. The strict verifier also widens stored bounds in the safe direction, encloses trigonometric values through an interval for π plus Taylor bounds instead of

trusting ordinary library calls, checks symmetry before reduction, and fails closed on node caps or unresolved terminal boxes. These checks are what authorize the model's candidate to persist; they are not generated confidence scores.

E2 also used the final decomposition to attribute the gain correctly. The frozen window has H = 0.672167187145431, below the window-only external reference value 0.672500703679 recorded in the archive. The 67.350003708785593% result therefore does not come from claiming a better window. It comes from the additional matrix/pressure correction made available by the finite-dimensional certificates. This attribution is part of the demo result: the model not only found a stronger witness, but isolated which mechanism actually supplied the improvement.

The frozen conclusion is therefore precise. VeriLoop E2 derived the research route, proposed and revised the finite-dimensional witness, and supplied the reasoning that led to the retained 67.350003708785593% construction. The team's internal trial Harness independently executed the exact and interval verification path, rejected non-closing candidates, replayed the required certificates, and froze the result only after the declared obligations closed. The resulting artifact is a strict finite-dimensional certificate whose exact assembly gives 67.350003708785593%. What has not yet been established is an accepted Lean challenge entry or a completed end-to-end formal normalization bridge into the upstream Zeta23 theorem. The demo must therefore be described as an E2-derived, Harness-verified finite-dimensional result—not as a proof of the Riemann Hypothesis and not yet as a new kernel-verified ledger record.

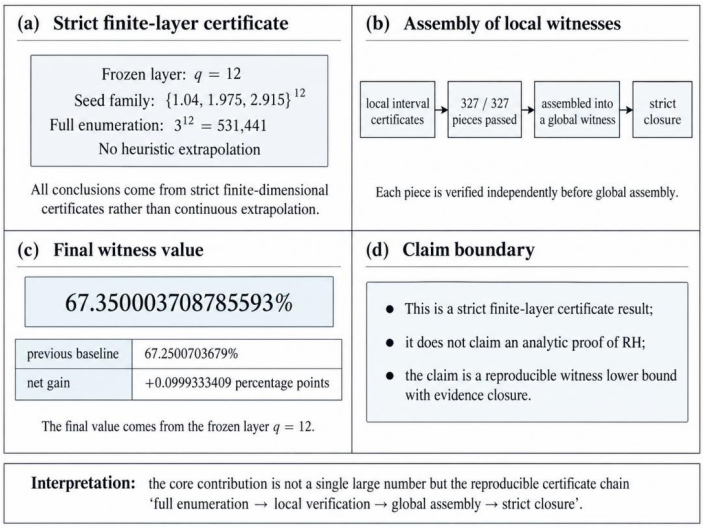


Figure 12 | Strict finite-dimensional certificate pipeline. The accepted q=8 witness is checked through three local inequalities, independently certified wells where applicable, fail-closed global complement or direct branch-and-bound, and exact rational assembly to 67.350003708785593%.

5.3.6 What this demo reveals about VeriLoop E2 reasoning

The significance of the Riemann demo is not that E2 eventually printed a larger decimal. The model repeatedly changed the representation of the problem in response to evidence. It abandoned a non-binding optimization direction, converted counterexamples into a richer seed geometry, distinguished mathematical stalls from numerical and storage stalls, redesigned the proof around certificate margin, and finally separated numerical gain from trust-boundary improvement. Each step changed what was proposed next because an external check had changed what was defensible.

Rejected trajectories were therefore productive without becoming persistent state. The refuted 67.350352% and 67.350063% candidates exposed missing geometry; the first-order table exposed an error floor that brute force could not beat; the positive-definiteness gate exposed a bound that was correct but practically inert; and the short tier exposed proof cost as a first-class optimization

variable. E2 used each failure to reformulate the next move, while the Harness preserved the last certified state.

Operationally, VeriLoop E2 acted as the proposal, abstraction, and diagnosis engine. It generated candidate witnesses, proposed seed geometries and verifier bounds, compared proof-cost tradeoffs, and redirected the search when counterexamples or enclosure diagnostics invalidated a route. The internal Harness acted as the persistence authority: it executed exact and interval checks, maintained the evidence contract, replayed certificates, rejected regressions, and froze only evidence-closed states. The result is therefore a system-level scientific-reasoning demo of E2, not a claim that the model's own confidence certifies mathematics.

The archive does not preserve a trustworthy start-to-finish wall-clock trace for the entire campaign, so the report does not invent one. What is reproducible is the decision chain itself: exact configuration data, strict local logs, independent well certificates, exact assembly, explicit refutations, and the rules governing which candidate states were allowed to survive. That chain is the dynamic capability being demonstrated: E2 does not merely answer a fixed question once; it revises a research program under externally enforced evidence.

5.4 Demo II: Asymptotic Graviton Tomography — A Verifiable Probe of the Black-Hole Information Problem

5.4.1 Demo setup: the black-hole information question and the first tractable boundary

The second demo uses the same VeriLoop E2 checkpoint as the model-side reasoning engine for a sharply delimited subproblem relevant to the black-hole information problem. The scientific question is how much information in free outgoing gravitational radiation at future null infinity can be reconstructed from an accessible observable algebra, and how robust that reconstruction remains at finite precision. The demo does not contain interacting black-hole dynamics, an evaporation model, or a solution of the black-hole information paradox. Its purpose is to test whether E2 can identify a mathematically controllable boundary, generate and revise theorem-level constructions, and push the analysis through counterexample checks, numerical certificates, and explicit scope control while the internal trial Harness independently governs what may be retained.

$$\mathscr{H}_N = \oplus_{r=0}^{N} \mathscr{H}_{r,N-r},$$

$N_R N$The inherited fixed-particle-number result decomposes the N-graviton Hilbert space by right-helicity count and identifies the commutant of the established gravitational flux algebra with the algebra generated by the sector label. Adding one bundle-covariant spin-4 Stokes coherence observable collapses that fixed-N commutant to scalars. Rather than treating this as a finished special construction, E2 asked what invariant in the proof actually caused the collapse.

The model's first abstraction was that the proof used only block connectivity. If an observable has a nonzero block between two sectors, it creates an edge; commutation forces the scalar coefficients on the two sectors to agree. From that observation E2 proposed a graph theorem for a family of added observables S: the exact commutant is the algebra of functions constant on connected components of the sector-connectivity graph $G(S)$. The Harness then checked the finite-dimensional instances and theorem-side algebra needed to retain that abstraction.

$$\mathrm{vN}(\mathrm{A}_0 \cup \mathrm{S})' \cong \ell^\infty \left(\pi_0\big(G(\mathrm{S})\big)\right).$$

$N{+}1NN$Informational completeness is therefore equivalent to connectivity within the stated sector model. The same abstraction immediately yields a resource lower bound: connecting all sectors requires at least a spanning-tree number of nonzero off-diagonal blocks, and the Stokes connector meets that bound. This is the first dynamic step of the demo: E2 converts a one-off construction into a reusable invariant, and the Harness decides whether the generalization survives exact checks.

5.4.2 Fock-space closure: choosing the connector that removes a hypothesis

The connectivity theorem changed a previous conclusion about the full Fock space

$$\mathscr{F}=\oplus_{N\geq 0}\mathscr{H}_N.$$

Because the existing flux and Stokes observables preserve total particle number, the number-sector graph has no edges. The full commutant is consequently

$$\mathrm{A}_{\mathscr{F}}^{'}=\mathrm{vN}(\widehat{N}).$$

not the scalars. This rules out an inappropriate shortcut to Stone-von Neumann irreducibility for the physically restricted algebra under study.

$\Phi(f)=a(f)+a^{\dagger}(f)$ To connect neighboring number sectors, E2 compared two natural proposals: a Weyl unitary and the field operator. The bounded Weyl proposal looked cleaner, but its adjacent blocks could become arbitrarily small and vanish at isolated parameter values, introducing an avoidable genericity assumption. E2 therefore rejected the superficially safer route and proposed the technically harder field operator, whose creation part obeys the exact injectivity identity below.

$$// a^{\dagger}(f)\psi //^2=// a(f)\psi //^2+// f //^2 // \psi //^2>0$$

$fa^{\dagger}(f)$Because the identity holds for every nonzero smearing function, the creation part is injective and every adjacent number-sector block is nonzero. The choice is important for the demo: E2 preferred the operator that removed a hypothesis from the theorem rather than the one that was easier to manipulate. The Harness then tested this prediction in truncated Fock-space calculations instead of accepting the argument on plausibility alone.

$N=0,\ldots,5N_{\max}+1f10^{-6}\Phi(f)f=0$ The executable checks reproduce the predicted structure. With the tested truncation, the commutant dimension before the linear connector is 6, exactly matching the six number sectors. For several nonzero choices of the connector, including a small-scale case, the commutant dimension falls to 1; the zero-control case remains at 6. The smallest measured singular value of the creation blocks in the reference check is 1.352775, strictly positive. These results promoted the connector choice from a model proposal to a retained demo state.

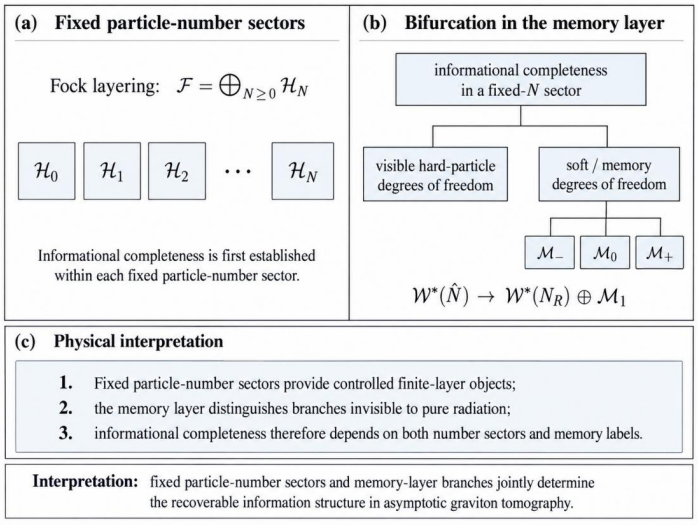


Figure 13 | Fock-sector and memory-layer structure in asymptotic graviton tomography. Fixed particle-number sectors provide the controlled block decomposition; the unresolved information is then localized to number-sector and memory-sector labels rather than being hidden in a generic "missing information" term.

5.4.3 Memory: refusing to build on a contested premise

$\omega \to 0$The next obstruction appears at the infrared boundary. The physical soft BMS charge has the right linear character to act as a connector, but the relevant mode is infrared non-normalizable and lies outside the one-particle Hilbert space assumed by the Fock theorem. E2 therefore reached a genuine boundary of the current proof rather than treating a formal analogy as a completed argument.

At this point the model could have forced a controversial premise—namely that memory sectors are superselected—into the derivation. Instead, E2 marked the literature disagreement itself as an unresolved protected coordinate and reformulated the result so that both branches could be handled. If memory sectors are disjoint, their projections are central, the inter-sector graph is empty, and no internal observable of the same algebra can create an edge. If the sectors are not disjoint, intertwiners supply edges and the same connectivity theorem classifies the resulting algebra. The Harness preserved the branch condition explicitly rather than letting the model hide it inside a stronger claim.

$1.33\times10^{-15}2.83\times10^{-16}$ The numerical certificate mirrors this logic. Three inequivalent model sectors produce a three-dimensional block-scalar commutant with only tiny numerical deviation from the predicted form, while a contrast experiment with three equivalent copies produces a nine-dimensional commutant containing off-diagonal intertwiners. Cross-sector phase dependence is numerically invisible to the tested algebra. The contrast case is essential: it shows that the no-go branch tracks sector inequivalence rather than an artifact of the implementation.

This step is a direct demonstration of E2's evidence-governed reasoning. The model did not "choose a side" in a disputed physical premise to keep the story moving. It changed the theorem so that the retained conclusion remained valid whichever branch the evidence eventually supports. The uncertain premise was removed from the commit condition instead of being buried in a caveat.

5.4.4 From exact-zero completeness to a sharp quantitative modulus

Exact connectivity still left a physically important question unanswered. A connected graph proves informational completeness in an exact algebraic sense, but it does not say how fragile that reconstruction is at finite precision. E2 treated this as a verifier-level objection to the previous state, not as a cosmetic limitation, and proposed rebuilding the argument around a weighted coherence-transfer Laplacian so that exact completeness and robustness could be measured by the same object.

For a sector-block observable S, define

$$w_{rr'}=\#P_{r'}SP_r\#_{\mathrm{HS}}^2, \qquad L=\mathrm{diag}\left(\sum_{r'} w_{rr'}\right)-(w_{rr'}).$$

For a block-scalar commutant candidate $X=\sum_r c_r P_r$, mutual Hilbert-Schmidt orthogonality of the off-diagonal blocks gives the exact identity

$$\#[X,S]\#_{\mathrm{HS}}^2=\sum_{r\neq r'}|c_r-c_{r'}|^2 w_{rr'}=2c^{\dagger}Lc.$$

The second eigenvalue $\lambda_2(L)$ is therefore not merely a connectivity diagnostic; it is the sharp modulus of completeness:

$$// c-\bar{c} // \leq \frac{// [X,S] //_{\mathrm{HS}}}{\sqrt{2\lambda_2(L)}}.$$

Equality is attained on the Fiedler eigenvector. The old exact criterion is recovered as the degenerate statement $\lambda_2>0 \Leftrightarrow G(S)$ connected. A separate operator-norm argument avoids Hilbert-Schmidt divergences in the continuum and yields

$$\mathrm{dist}(X,\mathbb{C}I)_{\mathrm{op}} \leq \frac{1}{2} D_R \; // [X,S] //_{\mathrm{op}},$$

where D_R is the resistance diameter formed from reciprocal inter-sector block norms.

This is the point where the demo moves from an exact yes/no statement to a quantitative law. E2 retained the old connectivity theorem as the zero/nonzero boundary of a stronger spectral-gap statement and proposed a second operator-norm route as an independent check. The Harness verified both derivations in their stated domains. The result is not merely "the algebra is complete," but a modulus that can be compared, stress-tested, and eventually confronted with more realistic dynamics.

5.4.5 Hidden su(2) structure and the unexpected *N*-independent gap

$N^{-2}N$E2 next asked how that quantitative modulus scales with graviton number. A uniform path-graph intuition suggested that the gap should shrink with N, making tomography progressively fragile. The numerical scan instead returned the same second eigenvalue at every tested N. Rather than freezing this as an empirical regularity, the model searched for a representation-theoretic mechanism that would force the constant.

On the two-mode reduction of the Stokes connector,

$$\mathrm{P}_\kappa = \lambda\left(a_R^\dagger a_L + a_L^\dagger a_R\right) = 2\lambda J_x.$$

At fixed N this is the spin-$N/2$ Schwinger-boson representation of $su(2)$. The adjoint action decomposes into spins $k=0,1,\ldots,N$, and the coherence-transfer Laplacian becomes one half of the adjoint Casimir on the J_z-commuting sector. Its spectrum is therefore

$$\mathrm{spec}(L) = \{\lambda^2 k(k+1) : k=0,1,\ldots,N\},$$

so that

$$\lambda_2(L) = 2\lambda^2$$

$$Nc_r = r - N/2N_R N$$

The calculation shows that the gap is exactly $2\lambda^2$ and independent of N in the stated two-mode reduction. The Fiedler vector is proportional to the same helicity-sector operator that the earlier commutant analysis had identified as the missing information. E2 thereby connected two results that initially appeared separate: the algebraic obstruction and the slowest coherence-transfer mode are the same direction viewed from exact and quantitative sides.

$N=3,5,8,12,20,40\lambda_2=2\lambda=19.1\times10^{-13}N=5,12,30,10010^{-14}\pi/\lambda N=4096\lambda=1\pi\approx3.141593$ The Harness then checked the spectrum across the reported finite-N range and verified the Fiedler vector independently; a separate operator-norm calculation produced the expected resistance-diameter behavior. These checks matter because the surprising N-independence was discovered numerically before it was explained. The demo retains the exact identity only after the model supplied a derivation and the independent checks agreed.

The exact gap also gives an operational resource law. If the connector channel is measured only to precision ε, commutators below that scale are experimentally unresolved and the residual ambiguity in the helicity-sector data is bounded by $\varepsilon/(2\lambda)$. Equivalently, resolving cross-helicity

coherence to accuracy δ requires detector precision on the scale $\varepsilon \approx 2\lambda\delta$. Within the two-mode su(2) reduction, this requirement is independent of graviton number N. E2 has therefore turned an abstract completeness statement into a quantitative measurement requirement.

$$// X - \bar{X} I //_{\mathrm{HS}} \leq \frac{\varepsilon}{2\lambda}, \qquad \varepsilon \simeq 2\lambda\delta.$$

The report keeps the domain boundary explicit. The coefficient $2\lambda^2$ is specific to the two-mode su(2) reduction of the Stokes connector. For a general multi-mode connector, the retained statement uses the actual $\lambda_2(L)$ or the dimension-free resistance-diameter bound. E2 does not export the special constant beyond the case the Harness can certify.

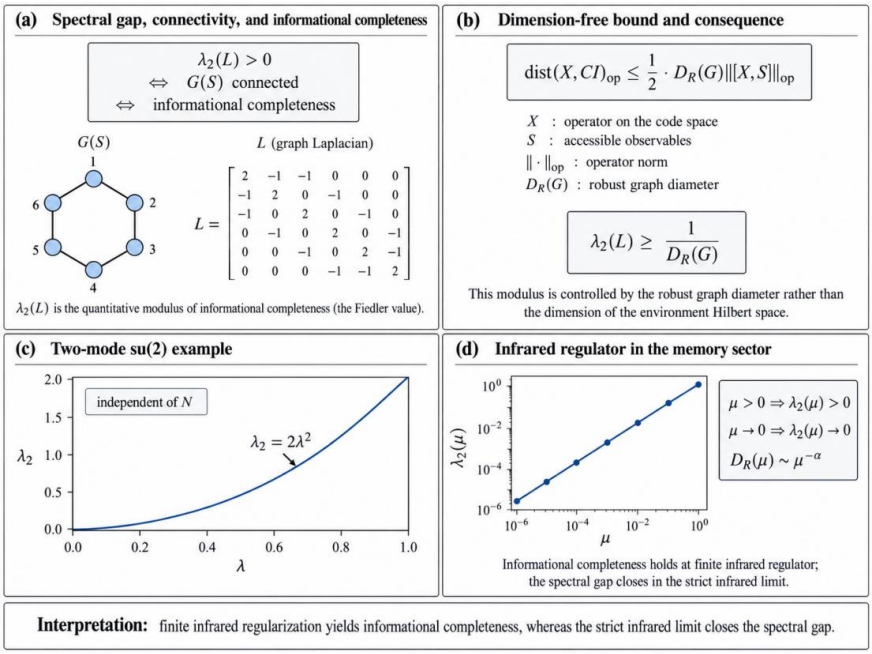


Figure 14 | Quantitative informational completeness. The second Laplacian eigenvalue simultaneously encodes graph connectivity and robustness; the two-mode Stokes connector has the exact gap 2λ2, while the infrared regulator drives the memory-sector gap continuously to zero in the strict infrared limit.

5.4.6 Turning the memory dichotomy into an order-of-limits statement

$\mu\lambda_2(\mu)>0$Once the spectral gap became the protected quantitative object, E2 revisited the memory-sector controversy without returning to a binary premise. With a finite infrared regulator, differently dressed sectors have nonzero overlap and the corresponding coherence weights remain positive; in the strict infrared limit the gap closes. This allowed the model to reinterpret the apparent disagreement as an order-of-limits problem rather than as a forced yes/no claim.

$$\mu>0 \Rightarrow \lambda_2(\mu)>0, \qquad \mu\to 0 \Rightarrow \lambda_2(\mu)\to 0.$$

$\mu^{\alpha}\mu^{-\alpha}\alpha$If the dressing overlap scales with the imported infrared exponent, the associated robustness modulus diverges at the corresponding inverse rate. The exponent itself comes from the dressing literature and is not derived by E2. What E2 contributes here is the structural identification of the spectral gap as the interpolating variable that carries the finite-regulator statement continuously into the strict limit.

λ_2 That reformulation is valuable because it is falsifiable. A future interacting treatment can alter the sector weights, graph, or scaling law, and the conclusion then changes through the measured spectral quantity rather than through an ambiguous label such as “memory is or is not

superselected." The demo therefore ends this branch with a stronger and more explicit claim surface than the one with which it began.

5.4.7 Certificate closure and the black-hole-information claim boundary

The internal Harness closes the asymptotic-graviton-tomography demo with six executable certificate groups. The reference run terminates with SUITE: 6/6 PASS. These checks cover the connectivity theorem, spanning-tree resource bound, Fock commutant and connector, memory dichotomy, quantitative modulus and exact su(2) gap, and a machine-readable scope guard that fails if the claims are weakened or broadened improperly.

Table 5.7 | Executable certificate closure for the asymptotic-graviton-tomography demo. The frozen reference suite reports 6/6 PASS.

Certificate	Main protected statement	Representative check
1	Sector Connectivity Theorem	Brute-force commutant dimension equals graph component count across single- and multi-operator cases
2	Spanning-tree resource bound	For four sectors, 0–2 nonzero blocks cannot complete; 3 can; the adjacent chain is edge-optimal
3	Fock commutant and linear connector	Truncated Fock commutant dimension 6 before the connector and 1 after any tested nonzero f; f=0 control remains 6
4	Memory dichotomy	Inequivalent sectors give a block-scalar commutant to 10^{-15} scale; equivalent-copy contrast produces intertwiners
5	Quantitative modulus and exact su(2) gap	Fiedler saturation ratio 1.0000000000 on random multi-mode families; $\lambda_2 = 2\lambda^2$ independent of tested N
6	Scope guard	Build fails if the machine-readable claim labels are weakened; reference run PASS

The retained theorem ledger marks T1, T1.1, T2, T3, T6, T7, and T8 as proved; T4 is retained as a dichotomy; T5 is conditional on the stated sectorwise structure; and T9 proves the interpolation structure while importing the infrared exponent. E2's role is the sequence of abstractions and derivations that produced these statements; the Harness's role is to keep their status and scope machine-checkable. Standard ingredients such as graph algebraic connectivity or the Schwinger-boson realization are not claimed as new.

Most importantly, this demo does not solve the black-hole information paradox. What VeriLoop E2 and the internal Harness establish is a chain of exact and quantitative results about free asymptotic radiative data: where the commutant resides, what kind of connector removes it, when memory creates a structural obstruction, what modulus controls approximate completeness, and why the specified Stokes connector avoids degradation with graviton number. Interacting dynamics, evaporation, and the map from initial-state information to the final asymptotic algebra remain outside the retained claim.

The demonstrated advance is therefore deliberately narrower but concrete. E2 upgrades the reconstruction problem from an exact connectivity criterion to a sharp quantitative completeness modulus; proves an N-independent gap for the specified two-mode Stokes connector; and reorganizes the memory-sector dichotomy as an infrared order-of-limits statement. These are the scientific outputs of the demo. They are evidence-closed advances toward understanding information recoverability at null infinity, not a unitary evaporation theorem.

5.4.8 What this demo reveals about VeriLoop E2 reasoning

Across the black-hole-information demo, the model's contribution is visible in a sequence of representation changes rather than in a single final answer. A special Stokes construction became a

connectivity theorem; an over-broad irreducibility intuition became an exact number-sector commutant calculation; a contested memory premise became a two-branch theorem; an exact-zero completeness result became a quantitative spectral-gap modulus; and a surprising numerical constant became an su(2) identity. Each change enlarged the domain of what could be defended or narrowed the claim when the evidence required it.

N^{-2}That progression is exactly the behavior the VGR design is meant to support. E2 is allowed to make aggressive proposals—memory superselection, Weyl connectivity, N-dependent degradation, or a constant numerical gap—but none becomes persistent merely because the model can argue for it. The internal Harness subjects each proposal to symbolic, numerical, contrast, or scope checks. When a route fails, the failure is retained as evidence that changes the next proposal; the last verified research state is not rewritten to make the narrative appear smoother.

In this demo, VeriLoop E2 functions as the reasoning engine and the internal trial Harness as the evidence governor. E2 performs the conceptual compression, generates alternative mathematical objects, diagnoses why a route is too weak, and proposes the next derivation. The Harness executes the checkable consequences—commutant dimensions, connector controls, spectral gaps, contrast experiments, theorem-status ledgers, and scope guards—and decides which candidate states can be frozen. The scientific result therefore emerges from model reasoning constrained by external evidence, rather than from either the model or the verifier acting alone.

As in the Riemann demo, the frozen artifacts do not preserve a reliable start-to-finish wall-clock trace for the entire campaign, so the report does not assign an unsupported hour count. What is reproducible is the reasoning-and-verification chain: theorem statements, abandoned routes, contrast cases, numerical certificates, machine-readable scope labels, and the final 6/6 passing suite.

Taken together, the two demos show the dynamic capability that benchmark tables cannot capture. The recorded model-side trajectories are bound to the VeriLoop E2 checkpoint: E2 proposes, abstracts, fails, diagnoses, reformulates, and continues across long technical trajectories. The internal trial Harness is equally essential but plays a different role: it executes externally checkable consequences, rejects non-closing states, preserves rollback integrity, and certifies only evidence-closed results. The demos therefore support a capability statement about E2 reasoning under Harness governance without converting model confidence into certification. The same attribution discipline applies to the benchmark campaign: checkpoint identity determines whose model behavior is measured, while the evaluation configuration determines whether a score is model-level or system-level. In that precise sense, the benchmark and demo program positions VeriLoop E2 not as already-complete verifiable intelligence, but as a concrete step toward it.

6 Conclusion, Limitations and Future Directions

6.1 Conclusion

VeriLoop E2 studies a narrow but consequential question: how a capable pretrained model can be adapted and deployed so that proposed progress is separated from authority to persist that progress. VGR answers this question with a fixed protected order, atomic retention, rejected-state isolation, and supervision derived from the same verifier semantics used at runtime. The VeriLoop E2 checkpoint supplies proposal, abstraction, diagnosis, and replanning; the Harness and verifier retain execution, rollback, and certification authority. This separation is the central design invariant across the architecture, training program, benchmark campaign, and scientific demos.

The empirical program supports that design at three complementary levels. First, the nine-track benchmark campaign shows broad capability across repository repair, terminal execution, mathematics, and science without collapsing heterogeneous tasks into a synthetic aggregate score. Second, the post-training evidence shows substantial first-to-last evaluation-loss reduction across all six general-capability surfaces while preserving an evidence-first recipe ledger that

distinguishes recorded fields from NOT_RECORDED metadata. Third, the two scientific demos expose dynamic behavior that endpoint benchmarks omit: failed routes generate evidence, verified components survive, claims narrow when assumptions cannot be closed, and stronger states are frozen only after independent checks complete.

The resulting conclusion is deliberately stronger than a model-card provenance statement but narrower than a universal correctness claim. VeriLoop E2 demonstrates that evidence-governed recurrence can be made operational as a post-training and systems discipline: learning can improve the distribution of proposals, while a non-learned contract decides which outcomes may become retained state. The formal guarantees remain contract-relative, and the benchmark evidence does not yet quantify causal uplift over the untouched Qwen 3.8-27B base. Within those boundaries, VGR provides a coherent route from verified trajectories to learned correction behavior without transferring verifier authority into model confidence.

6.2 Limitations

The first limitation is causal identification. The report binds all E2 results to the post-trained checkpoint, but it does not contain a matched raw-base-versus-E2 campaign under identical decoding, Harness, tool, and evaluator contracts. Consequently, the benchmark scores establish what the evaluated E2 checkpoint achieved, while the magnitude of improvement caused specifically by post-training remains unmeasured. This is a missing counterfactual, not a conflict with the recorded E2 results.

The second limitation is architectural integration. Section 3 specifies a latent VGR realization and a low-rank proposal operator, whereas the reported evaluations establish the checkpoint-plus-system trajectory realization. A native Qwen implementation must still demonstrate that the controlled latent is consumed end to end, dependent KV-cache state is recomputed or restored consistently, gradients reach the proposal operator during training, and the same interface is active at deployment. Until that integration evidence exists, the latent operator remains a reference architecture rather than an asserted hidden component of the reported runs.

The third limitation is verifier coverage. Protected non-regression is only as semantically strong as the fixed obligation schema. A zero rank certifies closure under that contract; it cannot prove behaviors the verifier never checks. The scientific demos make this boundary explicit—finite-dimensional certificate closure is not the Riemann Hypothesis, and asymptotic graviton tomography is not a unitary black-hole evaporation theorem. Scaling VGR to less formal tasks will require verifiers that are broad enough to matter while remaining deterministic, auditable, and resistant to reward hacking.

The fourth limitation is reachability under monotone persistence. VGR deliberately forbids a retained transition that worsens any protected coordinate, even when a globally successful solution path might require a temporary regression. Rejected proposals can inform later exploration, but the current theorem does not guarantee that an admissible direct route exists within the proposal radius or finite attempt budget. The mechanism therefore protects the incumbent at the cost of excluding some non-monotone retained trajectories.

The fifth limitation is evidence granularity and archival completeness. The frozen training package intentionally leaves several recipe fields NOT_RECORDED rather than reconstructing them. Separately, the long scientific campaigns do not preserve trustworthy start-to-finish wall-clock traces. These omissions do not alter the recorded state-transition or certificate evidence, but they prevent stronger claims about training reproducibility, compute efficiency, or total research-time acceleration. Future releases should capture model, verifier, recipe, timing, and environment identities in a single immutable run manifest from the start of each campaign.

6.3 Future directions: evidence-governed recurrent depth

A high-value next step is to turn VGR from a trajectory-supervision and external-persistence mechanism into a cache-consistent native recurrent-depth interface. Recurrent-depth language models already show that additional latent computation can improve reasoning [3], but recent work also reports a characteristic failure mode: performance can peak at an intermediate recurrence depth and then collapse as recurrent dynamics become unstable [20]. This is precisely where VGR supplies a missing systems primitive. Instead of asking only whether the hidden state converges, the next architecture should ask whether additional latent work produces an artifact-level transition that is externally defensible.

We propose an evidence-governed recurrent-depth program with two state planes. A certified plane holds the last retained latent–artifact–verification bundle and is immutable under failed attempts. A sandbox plane supports deeper recurrent exploration, including temporary regressions and multiple branches, without persistence authority. Candidate branches periodically materialize through a cache-consistent decoder; verifier receipts then supply strict-progress, regression, and incomparability labels. Only an admissible branch may replace the certified plane. This preserves VGR's non-regression guarantee for persistent state while relaxing the current reachability bottleneck inside private exploration.

The learning problem is then evidence-aware credit assignment over variable recurrent depth. Rather than supervising every loop with a token-level loss, the model can learn which hidden-state transitions eventually produce verifier-approved artifacts, with same-incumbent comparisons providing off-policy preference signals and the protected evidence vector providing structured credit. A practical implementation should jointly learn proposal direction, recurrence depth, and branch allocation while keeping the commit predicate non-learned. Cache invalidation and replay become part of the contract: any KV or recurrent state that depends on a changed latent must be recomputed or content-bound before verification, preventing stale-state artifacts from being certified against the wrong computation.

This direction directly targets the report's main limitations. It supplies the missing native latent integration, gives non-monotone search a safe private workspace, and addresses recurrent-depth instability with externally measured progress rather than internal convergence alone. The decisive experiment is also clear: train matched raw-base, ordinary post-trained, recurrent-depth, and evidence-governed recurrent-depth variants under the same data and verifier contracts; then measure benchmark capability, capability retention, admissible-commit rate, rollback frequency, recurrence-depth scaling, and compute per verified improvement. Such a study would convert the present architectural argument into a causal test of whether VGR improves the efficiency and reliability of latent reasoning itself.

The broader objective is not to prevent exploration, but to prevent exploration from silently rewriting what the system already knows how to defend. The next phase therefore moves VGR deeper into recurrent Transformer computation while preserving the same discipline: a learned model explores, external evidence governs persistence, and failed branches remain informative without becoming retained state. Success would not mean an "always correct" model; it would mean that additional internal computation is increasingly coupled to externally auditable progress.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017. https://arxiv.org/abs/1706.03762.

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

[3] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. arXiv:2502.05171, 2025. https://arxiv.org/abs/2502.05171.

[4] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. *International Conference on Learning Representations*, 2025. https://arxiv.org/abs/2410.08146.

[5] Tianyang Zhou, Somesh Jha, Mihai Christodorescu, Kirill Levchenko, and Varun Chandrasekaran. Verifier-Guided Code Translation via Meta-Step Decoding. arXiv:2605.17626, 2026. https://arxiv.org/abs/2605.17626.

[6] Tsinghua SIGS Robot Lab. VeriLoop Coder-E1. Public model card, accessed September 16, 2026. https://huggingface.co/tsinghua-sigs-robot-lab/veriloop-coder-e1.

[7] Tsinghua University. Tsinghua SIGS team releases the open-source code model VeriLoop Coder-E1. Institutional news release, August 12, 2026. Original in Chinese; title translated. https://www.tsinghua.edu.cn/info/1175/127603.htm.

[8] Kimi Team, Guangyu Chen, Yu Zhang, et al. Attention Residuals. arXiv:2603.15031, 2026. https://arxiv.org/abs/2603.15031.

[9] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-Depths: Dynamically Allocating Compute in Transformer-Based Language Models. arXiv:2404.02258, 2024. https://arxiv.org/abs/2404.02258.

[10] Bum Jun Kim, Kohei Hayashi, Shunsuke Kamiya, Masanori Koyama, Yusuke Iwasawa, and Yutaka Matsuo. Looped Transformers with Source-Centered State Evolution. arXiv:2607.27656, 2026. https://arxiv.org/abs/2607.27656.

[11] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. *Advances in Neural Information Processing Systems*, 35, 2022. https://arxiv.org/abs/2203.14465.

[12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations*, 2022. https://arxiv.org/abs/2106.09685.

[13] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, 36, 2023. https://arxiv.org/abs/2305.14314.

[14] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. *Advances in Neural Information Processing Systems*, 36, 2023. https://arxiv.org/abs/2305.18290.

[15] Anthropic. Zeta23 Lean 4 formalization and public challenge artifacts for critical-line zero-proportion bounds. 2026. https://github.com/anthropics/zeta-23-lean; https://www.riemannzeta.fun/.

[16] Miroslav Fiedler. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 23:298–305, 1973.

[17] Aldo Riello and Michele Schiavina. Null Hamiltonian Yang–Mills theory: soft symmetries and memory as superselection. Annales Henri Poincaré, 2024. arXiv:2303.03531.

[18] Laura Donnay and Yannick Herfray. Infrared physics of QED and gravity from representation theory. arXiv:2603.06297, 2026.

[19] Gordon W. Semenoff and Christopher Waterfield. The Wilson-line-dressed charged sector of scalar QED: superselection and the infraparticle. arXiv:2609.06224, 2026.

[20] Xiao-Wen Yang, Ziyu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing Recurrent Dynamics for Test-Time Scalable Latent Reasoning in Looped Language Models. arXiv:2605.26733, 2026. https://arxiv.org/abs/2605.26733.

Appendix A. Complete notation and symbol glossary

This appendix gives the exhaustive definitions, domains, and scopes of the mathematical symbols and operators used in Sections 3.1–3.12. Local-scope symbols are identified explicitly so that reused letters cannot be mistaken for globally shared variables.

Typographic convention. Single-letter mathematical variables and Greek variables are italic; multi-letter operators and implementation identifiers (for example, Digest, Bind, Initialize, Dequant, NLL, and certified) are upright roman. Subscripts that denote roles or labels (for example, lat, in, pref, act, gen, op, max, and train) are roman; subscripts that are mathematical indices remain italic. A prime marks the current candidate associated with an attempt; superscripts + and − mark positive and negative supervision examples, not arithmetic signs.

Symbol / operator	Precise definition, domain, and scope
x; idx; d_x	x is the task; idx is its nonempty task identifier; d_x = Digest(x) is the canonical task digest.
$\mathscr{R}$; d_M; ι_V; d_V; d_Γ	$\mathscr{R}$ is the fixed run contract. d_M is the model-contract digest, ι_V the verifier identifier, d_V the verifier-contract digest, and d_Γ the evidence-compiler-contract digest.
$O = (o_1, \ldots, o_m)$	O is the ordered obligation list fixed by the verifier contract; o_j is protected obligation j.
$I = (i_1, \ldots, i_m)$	I is the ordered list of distinct obligation identifiers; i_j identifies obligation o_j.
m	$m \in \mathbb{N}$ is the number of protected obligation coordinates in the fixed schema.
t	$t \in \{0,1,2,\ldots\}$ is the formal task-level proposal-attempt index used throughout the mathematical development.
k	k has local scope only. In Figure 2 it is a schematic attempt counter; in Equations (30)–(32) it is the target-token index; in 0_k it denotes a generic zero-vector length. These uses are local and are not Transformer-layer indices.
j	$j \in \{1,\ldots,m\}$ indexes protected evidence coordinates.
ℓ	$\ell \in \{1,\ldots,L\}$ indexes latent sequence positions in Equation (10); it is distinct from the log-likelihood symbol ℓ_ϕ.
A_x	A_x is the fixed task-conditioned source anchor for the run.
H_0; H_t; H_t'	H_0 is the baseline latent state; H_t is the retained latent at attempt t; H_t' is the candidate latent constructed for that attempt. Each lies in $\mathbb{R}^{L\times d}$ within a run.
L; d	L is the controlled latent sequence length and d is the hidden width; thus $H_t \in \mathbb{R}^{L\times d}$.
Δ_t	$\Delta_t \in \mathbb{R}^{L\times d}$ is the proposer-supplied latent displacement for attempt t.
E_t	$E_t \in \mathbb{R}^{L\times d}$ is exploratory memory containing the most recent rejected displacement; it is cleared after a successful commit.
$F_{\theta,\phi}$	$F_{\theta,\phi}$ is the learned proposal function that maps the task, anchor, retained state, rejected-displacement memory, evidence feature, and attempt index to Δ_t.
$\text{Initialize}_{\theta,\phi}$	Initialize, parameterized by the fixed model/adaptation snapshot θ and ϕ, is the task-conditioned initialization operator in Equation (2). Given task x, it returns the source anchor A_x and baseline latent H_0. The parameter subscript identifies the snapshot used for initialization; the operator does not update θ or ϕ inside a run.
ρ; τ_{lat}	$\rho > 0$ is the controller legality radius for each latent-position displacement; $\tau_{\text{lat}} \geq 0$ is the declared numerical tolerance used only in that legality check.
$\lVert\cdot\rVert_2$; max	$\lVert\cdot\rVert_2$ is the Euclidean norm; max denotes the maximum over the explicitly stated index set.
θ; ϕ	θ denotes backbone/model parameters for the latent realization; ϕ denotes trainable adaptation parameters. They are held fixed within a certified run snapshot.

Symbol / operator	Precise definition, domain, and scope
θ_0	θ_0 denotes the fixed pretrained parameter set used in the post-training likelihood equations; it is not updated by the loss terms shown.
$D_{\theta,\phi}$	$D_{\theta,\phi}$ is the fixed decoder used within a run; it materializes task artifacts from the controlled latent state.
y_t'; w_t'; ω_t'	y_t' is the candidate output sequence, w_t' its machine-checkable witness, and ω_t' its metadata, all decoded from H_t'.
a_t; a_t'	a_t is the retained bound artifact; a_t' is the candidate bound artifact. Each binds decoded content and the associated latent/model/task identity.
Bind; Digest	Bind is the canonical artifact-binding operation; Digest is the canonical digest function used to bind identities and bundle content. These names denote deterministic operators, not trainable variables.
V	V is the external verifier. It receives the task and a bound artifact and returns a structured verification result under the fixed verifier contract.
v_t; v_t'	v_t is the retained verification result and v_t' the candidate verification result at attempt t.
ζ; ζ_t; ζ_t'	ζ is the verifier diagnostic receipt in Equation (5); ζ_t and ζ_t' denote incumbent and candidate receipts used by the evidence compiler.
c	c in Equation (5) is the verifier completion flag. It is Boolean and is true exactly when the verifier reports completion under its declared contract.
r; r_t; r_t'	$r \in \mathbb{N}_0^m$ is a complete protected evidence-rank vector; r_t and r_t' are the incumbent and candidate ranks at attempt t. Smaller coordinates mean fewer remaining violations under the fixed obligation semantics.
r_j; $r_{t,j}$; $r_{t,j}'$	r_j is protected coordinate j; $r_{t,j}$ and $r_{t,j}'$ are its incumbent and candidate values.
$\preccurlyeq_P$; $\prec_P$; $\not\prec_P$	$\preccurlyeq_P$ is weak protected dominance (no coordinate worsens); $\prec_P$ is strict protected dominance ($\preccurlyeq_P$ plus at least one strict improvement); $\not\prec_P$ means the strict protected relation does not hold.
P	P labels the fixed protected partial order induced by the ordered obligation schema and verifier contract; P is not a learned score.
χ_t	$\chi_t = 1\{r_t' \prec_P r_t\}$ is the commit indicator: 1 authorizes atomic commitment and 0 requires retention of the incumbent bundle.
$1\{\cdot\}$	$1\{\cdot\}$ is the indicator function: it equals 1 when its predicate is true and 0 otherwise.
B_t; B_t'	$B_t = (a_t, H_t, v_t, \beta_t)$ is the complete retained CertifiedBundle; B_t' is the candidate bundle formed from a_t', H_t', v_t' and its recomputed digest.
β_t	β_t is the canonical binding digest of B_t, including task, source-anchor, artifact, latent, verification, and run-contract identity.
Φ; Σ / $\sum$	Φ is the integer evidence potential. The symbols Σ and $\sum$ denote summation. In the definition of Φ, the summation runs over protected-coordinate index j from 1 through m. In Equation (26), the summation runs over proposal-attempt index t across the finite run and counts strict commits through χ_t. Potential descent is only a proof/diagnostic consequence and never replaces the coordinate-wise commit predicate.
Γ	Γ is the fixed evidence compiler. It maps incumbent/candidate verification information and decision metadata to a bounded model-visible feature; it has no commit authority.
e_t	$e_t \in [-1,1]^q$ is the model-visible neural evidence feature at attempt t.
q; q_ζ	q is the total evidence-feature width; q_ζ is the width of the fixed receipt encoder ψ, with $q \geq 10 + q_\zeta$ in Equation (18).
κ	$\kappa > 0$ is the positive scale used to compress nonnegative rank magnitudes in the receipt-augmented summary.

Symbol / operator	Precise definition, domain, and scope
u; fu	u is a local nonnegative scalar argument used only in the bounded transform of Equation (17). f_u is that transform, mapping a nonnegative rank magnitude through the natural logarithm, scale κ, and tanh to a bounded scalar.
z; bz	The letter z has local scope and is reused in three explicitly separate contexts: in Equation (17), z is a Boolean input to b_z; in the definition of softplus in Equation (31), z is a real scalar input; and in Equation (21), z is a vector in $\mathbb{R}^d$ to which the radius projection is applied. For the Boolean use, b_z maps false/true to −1/+1 via the indicator function. These local uses are not the same variable.
$p_<$; $p_>$; $p_=$	$p_<$, $p_>$, and $p_=$ are the fractions of protected coordinates that improve, worsen, or remain unchanged from r_t to r_t'.
a; b; ht; ht,a:b	h_t is the ten-dimensional rank summary in Equation (17). The symbols a and b are local integer slice endpoints satisfying $1 \le a \le b \le 10$; $h_{t,a:b}$ denotes the inclusive component slice from index a through index b.
ψ	ψ is the fixed receipt encoder of width q_ζ used by Γ; it is contract-bound and has no persistence authority.
0_k; ;	0_k denotes a length-k zero vector. A semicolon inside a vector expression denotes concatenation, not punctuation between independent equations.
r_a	$r_a \le$ d is the proposal bottleneck width. It is a dimension parameter and must not be confused with evidence rank r or LoRA rank r_L.
W_s; W_a; W_e; W_u	W_s and W_a map hidden width d to bottleneck width r_a; W_e maps evidence width q to r_a; W_u maps r_a back to d. Their exact domains are given in Section 3.8.
LN_s; LN_a	LN_s and LN_a are affine-free outer LayerNorm operators applied to the state and anchor branches, respectively.
1_L; $\top$	1_L is the length-L all-ones column vector used for broadcast; $\top$ denotes matrix/vector transpose.
Z_t; T_t; U_t	Z_t is the fused bottleneck representation with real-valued shape $L \times r_a$; T_t is its residual-mixer output; U_t is the back-projected bounded-direction precursor with real-valued shape $L \times d$.
$\mathcal{M}$	$\mathcal{M}$ is the trainable low-rank residual mixer: affine LayerNorm → Linear(r_a, $2r_a$) → SiLU → optional dropout → Linear($2r_a$, r_a).
g; γ	g is the trainable scalar gate logit; $\gamma = \sigma(g) \in (0,1)$ is its sigmoid gate value.
σ; tanh; log; SiLU	σ is the logistic sigmoid; tanh, log, and SiLU are the standard hyperbolic-tangent, natural-logarithm, and sigmoid-linear-unit functions.
$\Pi_{M_{\mathrm{op}}}$	$\Pi_{M_{\mathrm{op}}}$ is Euclidean projection onto the closed radius-M_{op} ball, applied independently at each latent position.
M_{op}	$M_{\mathrm{op}} > 0$ is the proposal operator's internal projection radius. It is distinct from the controller legality radius ρ.
η; μ_t	$\eta > 0$ is the proposal step size; μ_t is the radius-bounded learned displacement mean before optional exploration noise.
ξ_t; σ_ξ	ξ_t is an optional finite exploration-noise tensor with the same shape as μ_t; $\sigma_\xi \ge 0$ is its scale.
N_{train}	N_{train} is the exact trainable-parameter count of the reference proposal module in Equation (24), excluding the backbone and any separate LoRA modules.
s_t; $T_{\max}$; $S_{\max}$	s_t is the number of consecutive rejected attempts; $T_{\max}$ is the finite attempt budget; $S_{\max}$ is the optional positive rejection-streak limit.
c_t	$c_t = 1\{r_t = 0\}$ is the retained zero-rank certification indicator in Equation (25). It is indexed by attempt and is distinct from the unindexed verifier completion flag c in Equation (5).
N_{commit}	N_{commit} is the total number of strict commits in a run; Proposition 2 bounds it by $\Phi(r_0)$.

Symbol / operator	Precise definition, domain, and scope
$S_{x,a}$	$S_{x,a}$ is an intended Boolean semantic-correctness predicate for task x and artifact a. Proposition 3 requires verifier soundness to infer $S_{x,a}$=1 from zero rank.
$r_{x,a}$	$r_{x,a}$ is the verifier rank assigned to artifact a for task x in the contract-relative completion statement.
$\mathscr{F}_n$	$\mathscr{F}_n$ is the complete history *σ-field/information set before the n*th idealized reachability attempt in Equation (28); n is local to that probabilistic argument.
h	$h \in \mathbb{N}$ is the local count of consecutive failed proposal attempts used only in the geometric tail bound following Equation (28).
Pr; ε	Pr denotes probability under the idealized proposal process; $\varepsilon > 0$ is the uniform lower bound on conditional strict-progress probability in Equation (28).
c_t^{in}	c_t^{in} is the model-visible conditioning record for post-training, built from the task, retained artifact, permitted evidence, and—when applicable—the accessible latent interface.
y^+; y^-; r^+; r^-	y^+ and y^- are same-incumbent positive-progress and noncommittable candidate sequences; r^+ and r^- are their verifier ranks.
y; n; yk; y<k	In Equations (30)–(32), y denotes the complete target sequence ($y_1,\ldots,y_n$); n is its target-token length; k is the local token-position index; y_k is the kth target token; and $y_{<k}$ is the prefix of target tokens preceding position k.
ℓ_ϕ; ℓ^+; ℓ^-	$\ell_\phi(y\|c_t^{\text{in}})$ is the teacher-forced sequence log-probability under trainable adaptation ϕ; ℓ^+ and ℓ^- are its values for y^+ and y^-.
$p_{\theta 0,\phi}$	$p_{\theta 0,\phi}$ is the conditional token distribution of the pretrained backbone θ_0 with trainable adaptation ϕ.
$\mathscr{L}_{\text{pref}}$; m_{pref}	$\mathscr{L}_{\text{pref}}$ is the pairwise preference loss; $m_{\text{pref}} > 0$ is the required positive log-probability margin.
softplus; e; ; pairs; Dact; D0	softplus is the standard softplus function used in Equation (31); its local input z is a real scalar. The unindexed e is Euler's number, the base of the natural logarithm, and is distinct from the evidence feature e_t. denotes mathematical expectation. The expectation subscripted "pairs" in Equation (31) is the empirical/minibatch expectation over valid same-incumbent comparison pairs (positive-progress versus noncommittable) checked under the same task and verifier contract and labeled by Equation (29). The two expectations in Equation (33) are empirical expectations over D_{act} and $D_{0,}$ respectively.
D_{act}; D_0	D_{act} contains verified strict-progress action records; D_0 contains accepted zero-rank final-artifact records.
NLL_ϕ	NLL_ϕ is the target-only masked negative log-likelihood used for action and final-generation supervision.
M_k	$M_k \in \{0,1\}$ is the supervision mask for target token k in Equation (32); prompt and diagnostic tokens have mask 0.
c^{in}	c^{in} in Equation (32) denotes the conditioning record for the corresponding supervised example. It is a local post-training symbol and is unrelated to the verifier completion flag c.
$\mathscr{L}_{\text{act}}$; $\mathscr{L}_{\text{gen}}$; $\mathscr{L}$	$\mathscr{L}_{\text{act}}$ is strict-progress action NLL, $\mathscr{L}_{\text{gen}}$ is zero-rank final-generation NLL, and $\mathscr{L}$ is the weighted total post-training objective.
λ_p; λ_a; λ_g	λ_p, λ_a, and λ_g are nonnegative weights on preference, action, and final-generation supervision, respectively.
W_0; W_ϕ	W_0 is a pretrained real matrix of shape $d_{\text{out}} \times d_{\text{in}}$; W_ϕ is its QLoRA-compatible adapted computational weight.
d_{in}; d_{out}	d_{in} and d_{out} are the input and output dimensions of the weight matrix in Equation (35). These role subscripts are roman.

Symbol / operator	Precise definition, domain, and scope
Q_4; Dequant	$Q_4(W_0)$ is the frozen four-bit representation of W_0; Dequant reconstructs the computational matrix used in forward evaluation.
A_L; B_L; r_L; α	A_L and B_L are trainable LoRA factors with shapes $r_L \times d_{\text{in}}$ and $d_{\text{out}} \times r_L$, respectively; r_L is the LoRA rank and α its scale. r_L is distinct from proposal bottleneck width r_a.
$\mathbb{R}$; $\mathbb{N}$; $\mathbb{N}_0$; $\in$	$\mathbb{R}$ denotes the real numbers; $\mathbb{N}=\{1,2,3,\ldots\}$ denotes the positive integers; $\mathbb{N}_0=\{0,1,2,\ldots\}$ denotes the nonnegative integers; $\in$ denotes set membership.
$\forall$; $\exists$; $\wedge$; $\Leftrightarrow$; $\Rightarrow$	$\forall$ means "for all", $\exists$ "there exists", $\wedge$ logical conjunction, $\Leftrightarrow$ logical equivalence, and $\Rightarrow$ logical implication.
$'$; $+$; $-$	A prime ($'$) on state/artifact/rank quantities marks the current candidate at the same attempt. Superscripts + and − in Section 3.11 label positive and negative supervision examples; elsewhere + and − retain their ordinary arithmetic meaning.
$=$; $\leq$; $<$; $\geq$; $>$; $\cdot$; /; $\times$	= denotes equality; $\leq$, $<$, $\geq$, and $>$ are the standard scalar order relations. A centered dot between operands denotes scalar multiplication, and / denotes scalar division. Within matrix/tensor domain annotations, × separates axis dimensions (rows by columns) rather than naming a learned operation. Coordinate inequalities in the protected order are interpreted only after schema and verifier-contract compatibility have been established.
$\mid$ / $\mid$	The vertical bar denotes conditioning, not division. In Equation (28) it separates an event from the history on which its probability is conditioned. In Equations (30)–(32) it separates the modeled target sequence or token from the conditioning record and, where shown, the preceding target prefix.
▫	▫ is the end-of-proof marker used after the proofs of Propositions 1 and 2. It has no algebraic value and is not a model, verifier, state, or optimization variable.