Graduated Dissent: Budgeted Disagreement Resolution for Multi-Model Inference

Andrew Michael Brilliant *

Posted Date: 24 March 2026

doi: 10.20944/preprints202603.1830.v1

Keywords: multi-model inference; disagreement resolution; error correlation; self-correction; formal verification; steelman exchange; budgeted inference; large language models

 doi:10.20944/preprints202603.1830.v1

Article

Graduated Dissent: Budgeted Disagreement Resolution for Multi-Model Inference

Andrew Michael Brilliant

Independent Researcher, Sapporo, Japan; a.brilliant@ieee.org

Abstract

Recent empirical work demonstrates that large language models cannot reliably self-correct reasoning without external feedback [2], and large-scale evaluation across hundreds of models reveals substantial error correlation even between models with distinct architectures and providers [4]. When generator and evaluator share failure modes, self-evaluation may provide weak evidence of correctness, and repeated self-critique may yield diminishing returns. External evaluation can address this, but external evaluation is expensive. A natural question arises: given a fixed verification budget, how should a system allocate costly decorrelated evaluation across queries? We propose *graduated dissent*, an inference architecture that treats this as a resource allocation problem. Multiple proposers generate candidate analyses in separated contexts. A comparator estimates whether divergence between proposals is superficial, within an expected domain noise floor, or structurally meaningful. Only high-signal disagreements trigger expensive procedures: steelman exchange, adversarial cross-examination, or external verification via formal proof checkers, executable tests, or numerical invariants. The proposed approach is budgeted inference: escalation occurs when the expected information gain from decorrelated evaluation exceeds its cost, given domain-calibrated priors on disagreement signal content. The protocol combines three mechanisms: context separation between generation and evaluation to reduce inheritance of error-producing reasoning traces, graduated triage to concentrate verification compute where decorrelation has the highest expected value, and a steelman exchange that encourages genuine engagement with opposing reasoning structures. We define domain-calibrated threshold structures and propose pre-specified benchmark families targeting technical reasoning reliability. This paper is a protocol proposal with pre-registered evaluation design; empirical results against the specified benchmarks will be incorporated in a subsequent version. The contribution complements existing evaluation approaches by providing an inference framework that may improve what reaches human judgment.

Keywords: multi-model inference; disagreement resolution; error correlation; self-correction; formal verification; steelman exchange; budgeted inference; large language models

1. Introduction

Large language models generate technically plausible analyses at low marginal cost, but reliability in technical reasoning may benefit from stronger evaluation alongside generation. In difficult reasoning tasks, including mathematical derivations, formal proof construction, and numerical reasoning with checkable invariants, the central challenge is often not producing an answer but determining whether that answer should be trusted. This becomes especially relevant in long-context workflows where intermediate assumptions accrete, local errors propagate, and repeated self-critique can consolidate confidence without necessarily adding independent evidence. Additional reasoning steps do not automatically constitute additional evidence. Decorrelated evaluation, when available, offers a complementary path forward.

Recent empirical work demonstrates that LLMs cannot reliably self-correct reasoning without external feedback [2]. Tsui [3] provides further evidence that this failure is not a knowledge deficit:

 doi:10.20944/preprints202603.1830.v1

models can correct identical errors when presented as external input but fail to correct those same errors in their own outputs, a phenomenon measured at a 64.5% failure rate across 14 models. An information-theoretic perspective offers one explanation [1]: when generator and evaluator share failure modes indexed by a latent variable Z, the information that self-evaluation provides about correctness may be bounded by $I(T; Z|G)$, which can approach zero when Z encodes only how the system fails, not whether it fails.

This motivates a practical question. If single-context self-evaluation has known limitations in certain regimes, what inference architecture might complement it in technical workflows? And given that decorrelated evaluation is expensive, requiring separate contexts, multiple models, or external tools, how should a system allocate that expense?

The approach proposed here is not maximal debate or brute-force ensembling. It is an inference framework in which disagreement between independently generated proposals is treated as a signal to be triaged rather than suppressed, and expensive verification is allocated where the expected information gain justifies the cost.

1.1. Relationship to Multi-Agent Approaches

Multi-agent LLM systems are an active and growing field. Many combine multiple model instances, personas, or debate rounds. A recurring challenge is that adding more voices does not automatically add more information. Identical or highly similar models may share failure modes, inherit the same context drift, and converge on the same answer with confidence. Always-on debate also introduces practical costs: increased latency, redundant critique, and token expenditure without proportional evidence gains.

The present work aims to complement these approaches in two respects. It draws on information-theoretic arguments [1] that formalize conditions under which self-evaluation may provide limited signal, offering criteria for when expensive decorrelated evaluation is most likely to be productive. It also treats disagreement resolution as a resource allocation problem with domain-calibrated thresholds, rather than applying uniform scrutiny to all queries or requiring premature consensus.

1.2. Contributions

The contributions are fourfold.

1. A **resource allocation formulation** of disagreement resolution under an error correlation framework, connecting graduated escalation to information-theoretic considerations about evaluation signal.
2. A **protocol** with three escalation levels, domain-calibrated thresholds, and a steelman exchange mechanism that encourages genuine engagement with opposing reasoning structures.
3. A **clean-context critique strategy** with concrete heuristics for reducing inheritance of error-producing reasoning traces, including state management mechanisms for long-session workflows.
4. **Pre-specified benchmark families** targeting technical reasoning reliability, specifically seeded derivation errors and long-context constraint retention, rather than generic hallucination detection.

1.3. Status of This Paper

This paper is a preprint. It presents a protocol design and pre-specifies evaluation methodology, but does not include empirical results. We publish it in this form deliberately: the benchmark specifications in Section 6 constitute a pre-registered evaluation protocol, documented before results are known, so that evaluation criteria cannot be adjusted post hoc. Comprehensive empirical validation is reserved for a companion paper implementing the specified benchmarks; formal submission of the present work is deferred until that empirical program is complete. Any deviations from the pre-specified evaluation design will be documented and justified. Because the empirical phase requires

 doi:10.20944/preprints202603.1830.v1

benchmark construction, adjudication, and verifier integration across multiple technical domains, we expect evaluation of the protocol to benefit from collaborative implementation.

1.4. Scope

We focus on technical tasks where correctness matters and verification criteria are at least partially available: mathematical derivations, formal proof subproblems, and numerical reasoning with invariants. Extension to scientific manuscript evaluation is a natural future direction but is outside the scope of the present paper. We do not claim generality to all conversational or creative settings. In domains without meaningful external checks, the architecture still offers process benefits, but its strongest guarantees weaken.

2. Theoretical Grounding: From Bounds to Architecture

This section develops information-theoretic bounds on self-evaluation [1] and extracts the architectural requirements they suggest. We present these as one motivating perspective among several for the design decisions, not as the sole source of validity. The architectural contributions of the present paper do not depend on the bounds being tight. The protocol would remain a useful inference framework even if the bounds were loosened, refined, or replaced by alternative arguments for decorrelated evaluation. The practical case for the architecture rests on three independent legs: the theoretical motivation summarized here, the empirical literature on self-correction limitations [2–4], and the standard ensemble diversity principle applied to the evaluation setting.

2.1. The Correlated Error Problem

Consider a generator producing hypothesis G for input X, and a selector producing evaluation score S. The correctness indicator is $T := \mathbb{I}\{G = H^\star(X)\}$. The central quantity is $I(T; S|G)$: the information that the selector provides about correctness, given that we already observe the generator output.

Theorem 1 (Information Bound; [1]). *Let Z be a latent variable indexing shared failure modes. Assume the conditional independence $S \perp T \mid (G, Z)$. Then:*

$$I(T; S \mid G) \leq I(T; Z \mid G) \tag{1}$$

In particular, if $T \perp Z \mid G$, then $I(T; S \mid G) = 0$.

Theorem 2 (Repeated Critique Bound; [1]). *Under the same conditional independence, k selector evaluations $A_1, \ldots, A_k$ satisfy:*

$$I(T; A_{1:k} \mid G) \leq I(T; Z \mid G) \tag{2}$$

That is, k critiques provide no more information about correctness than the single blind spot variable Z.

The implication is direct: under strong error coupling, self-evaluation may be non-identifying regardless of repetition. Additional passes through the same correlated evaluation pathway may yield diminishing returns. This is not a claim about any specific model's limitations; it is a structural property of evaluation systems where generator and evaluator share failure modes. Empirical evidence for the prevalence of such shared failure modes is provided by Kim et al. [4], who evaluated error correlation across over 350 LLMs and found that models agree on wrong answers 60% of the time on one benchmark dataset, with larger and more capable models showing higher error correlation even across distinct architectures and providers.

An important caveat: the conditional independence assumption $S \perp T \mid (G, Z)$ is strong, and the degree to which it holds varies across tasks and models. In some regimes, self-correction works adequately and the bounds are not tight. The architecture proposed here is most valuable in the regime where error coupling is high, which we expect to be common in technical reasoning with long

 doi:10.20944/preprints202603.1830.v1

dependency chains but which cannot be determined a priori for any specific query. This uncertainty is itself an argument for graduated rather than uniform escalation: the system should invest in decorrelated evaluation precisely when observable signals (disagreement, convergence failure) suggest the correlated regime may be operative.

2.2. The Resource Allocation Problem

The contrapositive of Theorem 1 identifies when evaluation *does* provide information: when the selector accesses information not contained in (G, Z), breaking the conditional independence. External selection channels, including formal verification, executable tests, different model families, and fresh-context evaluation, can satisfy this requirement.

But external evaluation is expensive. Formal proof checking requires compilation against a proof assistant. Executable verification requires test generation and execution. Even fresh-context critique requires additional inference passes. The system designer faces a constrained optimization:

> Given a fixed verification budget B, allocate decorrelated evaluation across queries $q_1, \ldots, q_n$ to maximize the total expected information gain about correctness.

Let ΔI_i denote the expected information gain from escalating query q_i to decorrelated evaluation, and let c_i denote the cost. The optimal policy solves:

$$\max_{\{e_i \in \{0,1\}\}} \sum_{i=1}^{n} e_i \cdot \Delta I_i \quad \text{subject to} \quad \sum_{i=1}^{n} e_i \cdot c_i \leq B \tag{3}$$

This is a knapsack problem. The objective is idealized: in practice, correctness probabilities are not known in advance. The protocol described in Section 4 is a practical approximation that estimates ΔI_i from observable disagreement structure and escalates when the ratio $\Delta I_i / c_i$ exceeds a domain-calibrated threshold.

2.3. Disagreement as Epistemic Signal

Why does disagreement between proposers serve as a useful proxy for ΔI_i?

When two proposers with partially independent failure modes agree, the joint probability of both being wrong is lower than either alone, so their agreement provides stronger evidence of correctness than either alone. When they disagree, at least one is wrong, and the disagreement localizes the region of reasoning where error may reside. The expected information gain from further investigation is higher.

Remark 1 (Design Principle: Multi-Proposer Advantage)**.** *If two proposers have partially independent failure modes, that is,* $\Pr(E_{S_1} \cap E_{S_2} \mid E_G) < \Pr(E_{S_1} \mid E_G)$*, then their joint acceptance provides stronger evidence about correctness than either alone. This is an application of the standard ensemble diversity principle to the evaluation setting. We state it as a design principle rather than a formal result because the degree of independence between proposers is an empirical quantity that depends on model selection, prompt design, and context management.*

Conversely, when proposers agree and both have partially independent error, the expected gain from further escalation is low. The system should accept synthesis. When they disagree substantively, the expected gain is high, and escalation is warranted.

The domain-calibrated noise floor (Section 4.5) distinguishes meaningful disagreement from expected variance. Not all divergence is signal. In literary analysis, substantial stylistic variation is expected; in theorem proving, small divergences may indicate proof failure. The noise floor captures this domain-specific prior.

2.4. From Bounds to Triage

The three levels of the protocol (Section 4) map directly onto the resource allocation framework:

 doi:10.20944/preprints202603.1830.v1

1. **Level 0 (Accept synthesis):** Estimated ΔI is low. Proposers agree within tolerance. Correlated self-evaluation would suffice; decorrelated evaluation would add cost without commensurate information.
2. **Level 1 (Flag with uncertainty):** Estimated ΔI is moderate but below escalation threshold. The system accepts the majority position but preserves the disagreement signal for human review.
3. **Level 2 (Escalate):** Estimated ΔI exceeds the threshold. The system invests in expensive decorrelated evaluation: steelman exchange, adversarial cross-examination, or external verification.

The threshold structure is domain-calibrated because the mapping from observable disagreement to expected information gain depends on domain-specific priors about error correlation, natural variance, and the cost structure of available verification tools.

3. System Design

The architecture is guided by five design goals motivated by the theoretical analysis: preserve diversity of analysis (premature consensus destroys the disagreement signal needed for triage), treat disagreement as potentially informative rather than suppressing it, reduce inherited context contamination by evaluating in cleaner contexts, spend verification compute selectively where expected information gain justifies the cost, and keep humans in command by concentrating attention on claims surviving progressively stronger scrutiny.

3.1. Functional Components

The architecture uses four functional components, defined in terms of the error correlation framework.

Definition 1 (Proposer). *A proposer P_i is a model instance that generates a candidate analysis R_i for task T under a bounded prompt specification, with failure modes indexed by blind spot set Z_i^{bad}. Two proposers are* partially decorrelated *if $Z_1^{\text{bad}} \neq Z_2^{\text{bad}}$, achieved through different model families, different prompt framing, or context separation.*

Definition 2 (Comparator). *A comparator J evaluates divergence between proposer outputs and estimates the expected information gain from escalation, classifying disagreement into escalation-relevant categories relative to a domain-calibrated noise model.*

Definition 3 (Verifier). *A verifier V is an external process whose evaluation criteria are not controlled by the proposers' training distribution, breaking the conditional independence $S \perp T \mid (G, Z)$ in the favorable direction. Examples include formal proof assistants, executable test suites, numerical invariant checkers, and retrieval-grounded source verification.*

Definition 4 (Arbiter). *The arbiter integrates proposer outputs, comparator judgments, and verifier results into a final recommendation for the human operator, with explicit uncertainty quantification and provenance tracking.*

A minimal deployment uses two proposers and one comparator, with verifiers invoked only on escalation. In terms of the resource allocation framework (Section 2.2), the comparator functions as an estimator of $\hat{\Delta I}_i$, the expected information gain from escalating query i to decorrelated evaluation. The SNR threshold (Equation (4)) operationalizes the decision rule: escalate when $\hat{\Delta I}_i / c_i$ exceeds a domain-calibrated threshold.

Remark 2 (Hardware heterogeneity and local deployment). *The multi-model architecture has a practical deployment advantage beyond its theoretical motivation: it maps naturally onto heterogeneous hardware. In local inference environments, relevant for practitioners operating under security, NDA, or data sovereignty constraints that preclude cloud APIs, different functional components can run on different accelerators matched to their*

 doi:10.20944/preprints202603.1830.v1

requirements. A high-memory GPU can serve the arbiter or large proposer model, while smaller models handle coding verification, Lean-based proof checking, or long-context retrieval tasks. This heterogeneous allocation can achieve higher aggregate utilization than a single large model, particularly when asynchronous batching allows heuristic evaluations and convergence monitoring to run concurrently with the primary reasoning workflow without blocking the main inference path.

3.2. The Comparator Problem

The comparator is the economically critical component: it determines whether additional compute is spent. It is also the least specified and arguably the most difficult part of the architecture. We elevate this to a first-class design problem rather than treating it as an implementation detail.

The core difficulty is that the comparator is itself likely a language model, and therefore may share the very blind spots the architecture is designed to address. If the comparator cannot reliably distinguish substantive dissent from cosmetic variance, the graduated logic becomes unstable: under-escalation accepts incorrect consensus, while over-escalation wastes budget on noise. The architecture partially mitigates this by asking the comparator to perform a narrower task than the proposers (classifying divergence rather than generating analysis), but the mitigation is incomplete.

In the domains within this paper's scope, we can partially constrain the comparator's task by anchoring divergence estimation in mechanically checkable features rather than relying entirely on semantic judgment. For formal mathematics and technical derivations, the comparator can check whether proposers reach the same final expression, whether they invoke the same lemmas or intermediate results, whether stated assumptions match, and whether derivation paths diverge at identifiable steps. These are structural comparisons that can be partially automated without requiring the comparator to assess correctness itself.

However, not all meaningful disagreements manifest as mechanically detectable divergence. Two proposers may reach the same final answer via derivation paths that differ in validity, or may agree on surface conclusions while disagreeing on unstated assumptions. Detecting these cases requires exactly the kind of semantic judgment that error correlation may compromise.

We identify three research directions for strengthening the comparator, none of which we claim to have solved:

1. **Structured comparison templates.** Domain-specific checklists that decompose comparison into mechanically answerable sub-questions (do the final expressions match? do the stated assumptions match? do the cited intermediate results agree?) rather than asking for a holistic similarity judgment.
2. **Comparator calibration.** Estimating the comparator's false negative rate (meaningful disagreement classified as noise) on held-out examples with known ground truth, and adjusting escalation thresholds to compensate. Conservative thresholds over-escalate, which wastes budget but preserves safety.
3. **Comparator decorrelation.** Running the comparator in a different model family from the proposers, or using an ensemble of comparators with diversity requirements analogous to those for proposers.

The honest assessment is that the comparator problem limits the architecture's reliability ceiling. In domains where divergence can be largely mechanized (formal proofs, numerical results), the limitation is mild. In domains requiring semantic judgment about reasoning quality, the comparator inherits some fraction of the evaluation fragility the architecture is designed to address. The graduated structure helps, because the comparator's task is classification (noise vs. signal) rather than verification (correct vs. incorrect), but the gap between these tasks is smaller than one might hope. To be explicit: this paper's primary claim is not that all components of the architecture are solved, but that this decomposition isolates the right control problem and yields a testable architecture whose individual components can be improved independently. We return to this in Section 8.

 doi:10.20944/preprints202603.1830.v1

3.3. Clean-Context Critique

A central design rule is that critique should not inherit the full reasoning trace that produced the original answer unless necessary. When the evaluating context inherits the generator's intermediate reasoning, error coupling increases and self-evaluation provides weak evidence. Context separation severs this inheritance: the evaluating context receives the candidate output (the conclusion, derivation, or claim) without the reasoning trajectory that produced it.

An important clarification: clean context means selective state transfer, not total amnesia. The evaluator still receives the problem statement, relevant constraints, and the candidate output. What it does not receive is the intermediate reasoning chain, the exploratory dead ends, and the accumulated framing that anchors the generation context. The goal is to remove the error-producing trajectory while preserving the information needed for meaningful evaluation.

Tsui [3] provides empirical evidence that even minimal disruption of the generation trajectory can substantially reduce error coupling: appending a single token that prompts the model to reconsider its reasoning reduced the self-correction blind spot by 89.3% without changing model weights. Full context separation is a stronger intervention, removing the error-producing trajectory rather than prompting reconsideration within it.

3.3.1. Practical Heuristics

Answer-Only Transfer.

Pass the candidate conclusion or derivation to the evaluator without the intermediate reasoning trace. This is the strongest form of context separation and is appropriate when the intermediate steps are not needed for evaluation.

Structured Summary Transfer.

When intermediate information is needed, pass a compact structured summary: stated assumptions, key equations or intermediate results, and the final claim. Omit exploratory dead ends and the narrative scaffolding that anchors the generation context.

Constraint Reinjection.

Reintroduce hard constraints explicitly at evaluation time rather than relying on their persistence from earlier in the context. Over long sessions, information stated at intermediate positions can be under-weighted or lost, as documented by Liu et al. [11]. Explicit reinjection ensures that evaluation occurs under the correct constraint set.

Role-Specific Framing.

Prompt evaluators with narrow, specific evaluation goals rather than general helpfulness. The default "helpful assistant" framing optimizes for diplomatic balance rather than rigorous assessment. Explicit critic framing ("identify all flaws," "find the weakest step") shifts the prediction target toward adversarial evaluation.

Checkpoint Compression.

For long-horizon workflows, replace accumulated context histories with compact checkpoints preserving only: active hard constraints, accepted intermediate results, unresolved disagreements, required verifications, and stop conditions.

3.3.2. State Management for Long Sessions

Extended reasoning sessions create a specific failure regime: constraints stated early or at intermediate positions may degrade in influence as context grows, notation conventions drift, and default model behaviors reassert [11].

The control plane maintains a lightweight state representation: active hard constraints, accepted intermediate results with provenance, unresolved disagreements, pending verification tasks, and

 doi:10.20944/preprints202603.1830.v1

explicit stop conditions. This state is injected into each new evaluation context, ensuring that critique operates under the correct constraint set regardless of how many reasoning steps have accumulated.

For constraints that must persist across extended sessions, the control plane supports periodic reinjection of anchor phrases: short, distinctive statements of critical constraints. A sliding-window deduplication mechanism checks whether each anchor is already present in recent context before reinjecting, avoiding unnecessary context budget expenditure.

4. Graduated Dissent Protocol

4.1. Protocol Overview

The protocol implements the resource allocation policy from Section 2.2 through a three-level escalation structure. The comparator performs divergence estimation and triage; the arbiter integrates outputs from all stages (proposers, comparator, steelman exchange, verifiers) to produce the final recommendation with provenance. In the algorithm below, synthesis operations labeled J represent comparator-level triage, while the final output A is the arbiter's recommendation.

Algorithm 1 Graduated Dissent Protocol

Require: Task T, proposers P_1, P_2, comparator J, verifier set $\mathcal{V}$
Ensure: Recommendation A with provenance and uncertainty
  {— Level 0: Independent proposal generation —}
  $R_1 \leftarrow P_1(T)$ {Proposer 1, context C_1}
  $R_2 \leftarrow P_2(T)$ {Proposer 2, context $C_2 \neq C_1$}

  {— Comparator triage —}
  $d \leftarrow J$.divergence(R_1, R_2)
  SNR $\leftarrow d \,/\, V_{\text{domain}}$ {Signal-to-noise ratio}

  if SNR $< \theta_{\text{accept}}$ **then**
    {Level 0: Agreement. Accept synthesis.}
    $A \leftarrow J$.synthesize(R_1, R_2)
    return A
  else if SNR $< \theta_{\text{escalate}}$ **then**
    {Level 1: Noise-floor disagreement. Accept with flag.}
    $A \leftarrow J$.synthesize_flagged(R_1, R_2)
    return A
  else
    {Level 2: Structural disagreement. Escalate.}
    {— Phase 1: Steelman exchange —}
    $S_1 \leftarrow P_1$.steelman(R_2) {P_1 constructs best case for R_2}
    $S_2 \leftarrow P_2$.steelman(R_1) {P_2 constructs best case for R_1}
    {— Phase 2: Self-critique of steelman —}
    $K_1 \leftarrow P_1$.critique(S_1) {P_1 critiques its own steelman of R_2}
    $K_2 \leftarrow P_2$.critique(S_2) {P_2 critiques its own steelman of R_1}
    {— Phase 3: Post-exchange comparison —}
    $d' \leftarrow J$.recompare$(R_1, R_2, S_1, S_2, K_1, K_2)$
    if $d' < \theta_{\text{resolve}}$ **then**
      $A \leftarrow J$.synthesize_post_exchange$(R_1, R_2, S_1, S_2, K_1, K_2)$
      return A
    else
      {— Phase 4: External verification —}
      Choose $V \in \mathcal{V}$ matched to disagreement type
      $E \leftarrow V$.verify$(R_1, R_2,$ disagreement locus$)$
      $A \leftarrow J$.final_synthesis$(R_1, R_2, S_1, S_2, K_1, K_2, E)$
      return A
    end if
  end if

 doi:10.20944/preprints202603.1830.v1

4.2. Level 0: Agreement Check

Compute fast similarity metrics and conclusion compatibility. If proposer outputs agree on conclusions, key intermediate steps, and stated assumptions within tolerance, accept synthesis. The rationale from Section 2.2: when partially decorrelated proposers agree, the expected information gain from further investigation is low. Their joint agreement provides stronger evidence than either alone (Remark 1).

4.3. Level 1: Noise-Floor Analysis

Compute a domain-calibrated signal-to-noise ratio:

$$\text{SNR} = \frac{\text{Semantic divergence}}{\text{Expected domain variance } V_d} \tag{4}$$

In practice, semantic divergence should be instantiated differently depending on domain and what is mechanically checkable. For the domains within this paper's scope, we recommend a composite measure combining: (i) conclusion comparison (do proposers reach the same final expression, numerical value, or proof result?), (ii) assumption matching (do proposers state the same premises, and do they invoke the same intermediate results?), and (iii) derivation path alignment (do the reasoning steps proceed through the same intermediate states?). Components (i) and (ii) are partially automatable via string matching, symbolic comparison, or structured output parsing. Component (iii) requires either manual annotation or an LLM-based comparator, subject to the limitations discussed in Section 3.2. The architectural contribution is the staged decision logic, not the specific divergence measure; but any deployment must commit to a concrete instantiation, and we recommend anchoring as much of the measure as possible in mechanically checkable features.

If $\text{SNR} < \theta_{\text{escalate}}$, accept the majority position with a flagged uncertainty note. The disagreement is preserved in the output for human review but does not trigger expensive verification. This level handles the common case: most disagreements between capable models are surface variation rather than structural conflict.

Convergence Monitoring.

As an optional extension, the SNR threshold need not be evaluated once. When multiple clean-context evaluations are available, the system can track inter-evaluation variance over successive rounds: decreasing spread suggests convergence toward a stable assessment, while persistent or oscillating spread indicates structural disagreement warranting escalation. This transforms triage from a single-shot threshold comparison into an iterative convergence test, with a cap of $k_{\max}$ monitoring rounds to prevent the monitoring process from becoming the cost sink it was designed to avoid.

4.4. Level 2: Adversarial Cross-Examination

By *structural disagreement* we mean disagreement traceable to different assumptions, interpretations, or derivation paths, not mere differences in exposition, notation, or step granularity. Operationally, within the domains in scope, structural disagreement is indicated when proposers reach different final results (different expressions, different numerical values, contradictory truth claims), when they invoke mutually exclusive assumptions, or when their derivation paths diverge at an identifiable step such that at least one path contains a logically invalid inference. Disagreements that reduce to notational variants, alternative but equivalent step orderings, or different levels of intermediate detail are classified as cosmetic. This distinction is not always sharp, and borderline cases are one reason the protocol includes a Level 1 category (flag with uncertainty) rather than forcing binary classification. Structural disagreement indicates that at least one proposer has made a substantive error or that the problem admits genuinely distinct solutions. This is where expensive verification has the highest expected information gain.

 doi:10.20944/preprints202603.1830.v1

When SNR exceeds the escalation threshold, the system invests in expensive decorrelated evaluation through the following phases.

4.4.1. Phase 1: Steelman Exchange

Each proposer constructs the *strongest possible case* for the opposing position. This element of the protocol deserves careful justification.

Standard debate asks each participant to argue for its own position and against the opponent. This can degenerate into defensive critique: the model generates counterarguments from its own perspective without genuinely engaging with the opposing reasoning structure. The steelman requirement inverts this. Each proposer must reconstruct the opponent's argument in its strongest form, which requires understanding the reasoning that produced the disagreement rather than merely attacking its conclusion. Operationally, a steelman must reconstruct the opponent's assumptions, inferential steps, and strongest defense of the disputed claim, not merely summarize arguments for and against.

This connects to the adversarial collaboration literature [12,13]: structured disagreement is more productive when participants are required to engage with opposing reasoning at the level of assumptions and structure, not merely conclusions. In the LLM context, the steelman forces the model to activate different prediction targets, predicting what a defender of the opposing view would say, which can partially decorrelate evaluation from the model's default reasoning trajectory.

Remark 3 (Steelmanning versus RLHF-trained hedging)**.** *The steelman exchange should not be confused with the balanced-summary behavior that RLHF-trained models produce by default. A model optimized for helpfulness will typically generate a surface-level survey (several points in favor, several against, hedged conclusion) without genuinely working through the opposing reasoning structure. The steelman exchange requires full reconstruction of the opposing argument as if the model were advocating for it, activating a different prediction target than neutral summarization. The distinction is analogous to the difference between reading a summary of a chess strategy and playing a full game from that strategy's perspective.*

4.4.2. Phase 2: Self-Critique of Steelman

Each proposer then critiques *its own* steelman of the opposing position. This serves two functions. First, it tests whether the steelman was genuine or merely a weak reconstruction. Second, it forces each proposer to articulate the strongest objections to the position it just defended, which can elicit assumptions that neither original proposal made explicit.

4.4.3. Phase 3: Post-Exchange Comparison

The comparator re-evaluates divergence after the steelman exchange. If the exchange resolves the disagreement (proposers converge after engaging with each other's reasoning), synthesize. If disagreement survives steelmanning, it is likely structural, reflecting genuinely different interpretations, assumptions, or derivation paths, and warrants external verification.

4.4.4. Phase 4: External Verification

If disagreement persists after cross-examination, the system invokes an external verifier matched to the disagreement type. The verifier provides evaluation criteria not controlled by the proposers' training distribution, breaking the error coupling that limits self-evaluation.

Remark 4 (Steelman as epistemic investment)**.** *Beyond decorrelation, the steelman exchange corrects an asymmetry: a model that has generated argument A has invested prediction effort in A's reasoning structure but has never inhabited position B's reasoning structure. Asking the model to "consider both sides" does not correct this; surface-level counterargument generation requires less engagement than full reconstruction. The steelman forces equivalent cognitive investment in both positions. No operationalized metric for steelman quality currently exists; the benchmarks in Section 6 are designed in part to measure whether steelmanning produces measurable improvements in disagreement resolution.*

 doi:10.20944/preprints202603.1830.v1

4.5. Domain-Calibrated Thresholds

Disagreement must be interpreted relative to the domain. In theorem proving, small divergences can indicate proof failure. In exploratory reasoning, greater variance is expected. The comparator requires domain-specific noise models.

Remark 5. *Table 1 covers the domains within this paper's scope and benchmark plan. The architectural principle, that acceptance and escalation sensitivity must vary by domain, extends naturally to other settings, where policies would be progressively more permissive. We defer calibration of those domains to future work where appropriate benchmarks exist.*

Table 1. Disagreement policies for domains within scope. Acceptance strictness and escalation sensitivity vary by domain characteristics. Specific numerical thresholds require empirical calibration per deployment.

Domain	Policy	Rationale
Formal mathematics / proofs	Strict accept, narrow noise floor	Small divergences can indicate proof failure; formal verification available as escalation target
Technical derivations	Strict accept, narrow noise floor	Sign errors, dropped terms, and hidden assumptions matter; numerical invariants available
Numerical / physics reasoning	Strict accept, moderate noise floor	Precision critical; strong formal constraints but some methodological variation expected

Calibration Procedure.

The noise floor V_d and escalation threshold θ_{escalate} are empirical parameters that vary by domain and proposer pair. The default deployment mode uses conservative thresholds (low θ_{escalate}, biasing toward over-escalation). This is operationally safe: the system may spend more verification budget than necessary, but it does not miss errors. Over-escalation wastes compute; under-escalation misses failures. The conservative default accepts the former to avoid the latter.

When labeled calibration data with known ground truth is available, thresholds can be tightened to reduce unnecessary escalation. The calibration procedure is as follows: (1) run both proposers independently on each calibration task, (2) compute the divergence measure for each task, (3) partition tasks into those where both proposers are correct (agreement-correct), those where both are wrong in the same way (agreement-incorrect), and those where they disagree (at least one wrong). The noise floor V_d is estimated from the divergence distribution of agreement-correct tasks: this is the variance the system should treat as non-informative. The escalation threshold θ_{escalate} is then set such that a target fraction (e.g., 90%) of agreement-incorrect cases would have been escalated, accepting the resulting false escalation rate on agreement-correct cases as the cost of safety. This calibration procedure also serves as a diagnostic for the comparator problem identified in Section 3.2: the false negative rate of the comparator on calibration data directly informs threshold setting, with a high false negative rate requiring a lower θ_{escalate} to maintain safety at the cost of increased compute.

In summary, the system degrades gracefully: without calibration data it over-escalates (safe but expensive), and with calibration data it converges toward efficient allocation. No domain is locked out; calibration is an optimization, not a prerequisite.

 doi:10.20944/preprints202603.1830.v1

5. Targeted External Verification

5.1. Verification as Escalation, Not Default

External verification provides a strong form of decorrelated evaluation: the verifier's acceptance criteria depend on ground truth (mathematical, computational, physical) rather than on patterns in training data. In terms of the error correlation framework, external verification breaks $S \perp T \mid (G, Z)$ in the favorable direction, because the verifier accesses information about correctness that is not contained in the proposers' shared blind spot variable.

However, verification is expensive. Lean compilation requires type-checking against Mathlib. Executable tests require generation and execution. Even retrieval-grounded fact-checking requires corpus queries and comparison. The protocol therefore invokes verification selectively: only when disagreement survives steelmanning and the expected information gain justifies the cost.

5.2. Verification Modalities

Different tasks support different external checks, ordered by decreasing strength of decorrelation.

Formal Verification.

Proof assistants (Lean, Coq, Isabelle) verify derivation fragments against mathematical ground truth. A proof that compiles is verified by mathematics itself. This provides a strong form of decorrelation: the verifier's acceptance depends on logical consistency, not on any property of the training distribution. The Prover-Agent framework [5] demonstrates this approach in practice, achieving strong results on MiniF2F using small models augmented with Lean feedback.

Executable Verification.

For code or algorithmic claims, unit tests and property-based tests provide selection under computational ground truth. The interpreter does not care how confident the model was; the code runs or throws an error. This may contribute to the observed effectiveness of LLMs at coding tasks: the code interpreter provides built-in external selection.

Numerical Invariant Checking.

Physical or mathematical invariants, including dimensional consistency, conservation laws, symmetry constraints, and sanity bounds, can falsify plausible but invalid reasoning. A derivation that violates energy conservation fails regardless of how convincing it sounds.

Retrieval-Grounded Source Verification.

For manuscript claims, citations and quotations can be checked against a fixed corpus. Exact quote attribution and fact-checking against authoritative sources provide selection under documentary ground truth.

5.3. Verification Dispatch

Let D denote disagreement class after steelman exchange and let c_V denote the cost of verification modality V. The routing policy selects the cheapest verification modality sufficient to resolve the disagreement:

$$V^* = \arg\min_{V \in \mathcal{V}} c_V \quad \text{subject to} \quad V \text{ is applicable to disagreement type} \tag{5}$$

If the disagreement concerns a mathematical derivation and Lean is available, route to formal verification. If it concerns a numerical claim with checkable invariants, route to numerical checking. If no matched verifier exists, the system reports the unresolved disagreement with full provenance for human judgment.

 doi:10.20944/preprints202603.1830.v1

6. Benchmarking Strategy

The benchmark specifications in this section are proposed protocols, not implemented evaluations. We publish them in this preprint deliberately as a pre-specified evaluation protocol: by documenting the intended evaluation design, including construction methodology, evaluation conditions, metrics, and predictions, before implementation and before results are known, the specifications ensure that evaluation criteria cannot be adjusted post hoc to favor the architecture, that conditions or metrics producing unfavorable results cannot be quietly dropped, and that the community can independently implement the same benchmarks to verify or challenge our results. This preprint will be updated with empirical results against these specifications prior to journal submission; any deviations from the pre-specified design will be documented and justified. Benchmark code and data will be made available at https://github.com/andrewbrilliant/graduated-dissent.

6.1. Complementing Existing Evaluation Benchmarks

Existing hallucination benchmarks provide valuable baselines for general factual reliability. The present benchmarks complement these by targeting a specific subset of technical reasoning failure modes where context drift, hidden assumptions, and evaluator coupling are most relevant. By narrowing the evaluation target, we aim to measure whether the graduated protocol provides value in the regime it is designed to address.

6.2. Benchmark Family A (Primary): Seeded Derivation Errors

We specify a benchmark of derivations containing controlled, single-point errors suitable for measuring error detection across evaluation architectures.

Construction Protocol.

Select 200 correct derivations spanning three difficulty tiers: undergraduate (calculus, linear algebra, classical mechanics), graduate (real analysis, electrodynamics, statistical mechanics), and research-adjacent (variational methods, group theory applications, renormalization arguments). Sources include standard textbooks and publicly available problem sets with known solutions. For each correct derivation, construct four erroneous variants by seeding exactly one controlled error:

(a) **Sign error:** flip a sign in an intermediate step (e.g., $+\partial_\mu \to -\partial_\mu$).
(b) **Unjustified substitution:** replace a variable or expression with a related but incorrect one (e.g., substitute a Taylor expansion truncated at the wrong order).
(c) **Dropped term:** remove a term that contributes to the final result, preserving surface plausibility.
(d) **Domain violation:** introduce a step that violates a constraint (e.g., divide by a quantity that can be zero, exchange limits without justification, apply a theorem outside its domain of validity).

This yields 800 erroneous derivations plus 200 correct controls, totaling 1000 instances. Each instance is labeled with error type, error location (step number), and difficulty tier. The correct derivations serve as a false positive control: the system should accept them without spurious objections. The system under evaluation is not informed whether a given derivation contains an error, nor which error type was seeded; correct and erroneous instances are presented in randomized order.

Evaluation Conditions.

Compare error detection rates across four conditions:

1. Single-model long-context chain-of-thought (baseline)
2. Same-model fresh-context critique (context separation only)
3. Multi-model protocol without external verification (protocol without tools)
4. Multi-model protocol with targeted external verification (full system)

 doi:10.20944/preprints202603.1830.v1

Metrics:

False acceptance rate (erroneous derivations accepted as correct), false rejection rate (correct derivations flagged as erroneous), error localization accuracy (whether the system identifies the correct step), and compute cost (total tokens per instance).

Predictions.

The information-theoretic analysis predicts that condition (1) will show the highest false acceptance rate, with diminishing returns from additional self-critique passes within the same context. Fresh-context critique (2) should improve over (1) by breaking context inheritance. Multi-model dissent (3) should further improve by introducing partially independent failure modes. External verification (4) should provide the largest gains on formally verifiable errors, particularly domain violations where invariant checking applies. We also predict that error localization will improve more sharply than detection rate across conditions, as the graduated protocol forces explicit identification of disagreement loci.

6.3. Benchmark Family B (Secondary): Constraint Retention Under Long Context

This secondary benchmark measures whether evaluation architectures preserve hard constraints across extended reasoning sequences. The design is preliminary and we invite community feedback via the project repository.

Tasks consist of a constraint declaration (numerical invariant, definitional rule, methodological restriction, or notational convention) stated in the first 5% of the context, followed by 30–60 reasoning steps that progressively introduce plausible opportunities for constraint violation. Constraint retention is compared across three conditions: single-model continuous context (baseline), single-model with periodic anchor phrase reinjection, and the proposed protocol with checkpoint-based state management. Primary metrics are constraint violation rate as a function of context depth and latency to violation detection.

6.4. Manuscript Claim Evaluation (Future Work)

A natural extension is evaluation of methodological or technical claims extracted from scientific manuscripts. One particularly clean evaluation design uses retracted papers published after model training cutoff dates: the ground truth is known (the retraction notice identifies the flaws), memorization is excluded by construction, and the task directly tests whether decorrelated evaluation surfaces problems that single-context evaluation misses. The system would analyze manuscripts blind to the retraction, and its flagged concerns would be compared against the actual reasons for retraction. This design provides a natural experiment in which the proposed architecture faces exactly the conditions it claims to address: hidden errors in plausible, internally coherent technical reasoning.

However, manuscript evaluation introduces substantial additional complexity, including labeling, detectability definitions, domain specificity, and the distinction between methodological errors and fraud, that warrants separate treatment and is outside the scope of this paper. The companion benchmark paper will additionally evaluate prompt architectures for structured adversarial review, including role-specialized evaluation agents with preloaded domain context.

6.5. Evaluation Metrics

Across all benchmark families, we track four metric classes.

1. **Reliability.** Error detection rate and false acceptance rate across conditions.
2. **Resolution quality.** Whether escalation improves outcomes on the queries that trigger it. Measured by comparing accuracy of the system's post-escalation output on Level 2 queries against the accuracy that Level 0 forced synthesis would have produced on those same queries (e.g., majority vote of initial proposers). This directly measures the value added by the escalation investment.

 doi:10.20944/preprints202603.1830.v1

3. **Calibration.** Whether system confidence tracks actual correctness. Measured by calibration curves and Brier scores.
4. **Efficiency.** Reliability gain per unit of compute. The primary measure is (accuracy gain)/(additional tokens or cost) relative to single-model baseline. To support optimization of individual protocol components, benchmarks will additionally log: total token count per instance across all model calls, wall-clock latency from query to final recommendation, total compute time across all parallel and sequential processes, and per-component breakdowns (proposal generation, comparison, steelman exchange, verification). These fine-grained logs allow practitioners adopting any subset of the protocol to identify cost bottlenecks and optimize selectively.

7. Design Motivations

The protocol design was motivated by practical observations from the author's use of multi-model evaluation in a theoretical physics research workflow. These are not controlled experiments and do not constitute empirical validation. We note them briefly for transparency about the practical experience that shaped the design.

Three patterns recurred across sessions. First, the majority of divergences between proposers were cosmetic (notation, step ordering, detail level), suggesting that the graduated protocol's primary practical value may lie in *not* escalating. Second, the steelman exchange frequently surfaced tacit assumptions that neither original proposal had stated explicitly, an assumption-surfacing function potentially as valuable as error detection. Third, outputs declared "ready" within a long generation context were frequently identified as flawed when evaluated in a fresh context, consistent with the theoretical prediction that context separation reduces error coupling. Rigorous validation of these observations requires controlled experiments across the benchmark families defined in Section 6.

8. Failure Modes of the Architecture

The architecture can fail. Enumerating failure modes explicitly is both intellectually honest and practically useful for deployment.

Correlated Proposers.

If proposers share training data, architecture, and prompt framing, they may disagree only cosmetically while sharing the same structural blind spots. The graduated protocol would classify their agreement as Level 0 (accept synthesis) when both are wrong in the same way. Mitigation: use different model families, different prompt framings, or maximally different context structures.

Comparator Bias.

As discussed in Section 3.2, the comparator is itself a language model and can misclassify meaningful dissent as noise (under-escalation) or surface variance as structural disagreement (over-escalation). Under-escalation is the more insidious failure: the system accepts incorrect consensus. This is not merely a failure mode but a fundamental limitation of the architecture; see Section 3.2 for mitigation strategies and honest assessment of residual risk.

Verification Mismatch.

External verification tools may be weak, spoofable, or badly matched to the disagreement type. Tests that don't actually test the claimed property provide false reassurance. More fundamentally, available verifiers may cover only a fraction of the error space: a sign error deep in a derivation may not violate any checkable invariant such as dimensional consistency or conservation laws. The architecture's strongest guarantee (external verification breaks error correlation) applies fully only when a matched verifier exists. For errors outside verifier coverage, the system falls back to the steelman exchange and comparator judgment, which provide weaker evidence. Mitigation: match verification modality to disagreement type, prefer hard external criteria (formal proofs, physical

 doi:10.20944/preprints202603.1830.v1

invariants) over soft checks, and report explicitly when no matched verifier is available rather than providing false confidence.

Excessive Escalation.

Overly sensitive thresholds cause cost blowups without proportional reliability gains. The system spends its verification budget on noise rather than signal. Mitigation: domain calibration and monitoring of escalation-rate-to-resolution-quality ratios.

Human Confirmation Bias.

The human operator may preferentially accept candidates that confirm prior beliefs, reintroducing correlated error at the final selection stage. Mitigation: blind review protocols, explicit checklists, and adversarial framing of human review prompts.

Steelman Collapse.

Models may produce weak steelmans, superficially engaging with the opposing position while actually reinforcing their own conclusion. A related failure is that a model may produce a steelman that is structurally indistinguishable from a sophisticated hedge, engaging with the opposing position at the level of rhetoric without genuinely reconstructing its reasoning. Distinguishing genuine steelmans from sophisticated hedges is itself a judgment that the comparator must make, and no operationalized metric for steelman quality currently exists. This is an open problem. Mitigation: explicit steelman quality checks, comparison of the steelman's reasoning structure against the original proposal's structure, and treating steelman collapse as a signal to escalate to external verification rather than accept synthesis.

9. Related Work

LLM Self-Correction.

Huang et al. [2] provided evidence that LLMs struggle to self-correct reasoning without external feedback. Tsui [3] measured a self-correction blind spot at 64.5% across 14 models and showed that minimal context perturbation can reduce it by 89.3%. The present architecture draws on information-theoretic arguments about correlated evaluation failure [1] as one motivation for the protocol design.

Self-Consistency.

Wang et al. [6] showed that sampling multiple reasoning paths and taking the majority answer improves accuracy. The present work complements this by additionally triaging whether agreement is informative given the degree of error correlation between sampled paths, rather than treating all agreement as equally evidential.

Ensemble Methods.

Traditional ensemble methods aggregate outputs to reduce variance, and have proven highly effective in many settings. The present approach extends this principle by additionally allocating scrutiny conditional on disagreement structure, which may provide further gains in the regime where errors are systematic and shared across models from similar training distributions. Kim et al. [4] provide large-scale empirical evidence for this regime, showing that LLM errors are substantially correlated and that correlation increases with model capability, even across different architectures and providers.

Multi-Agent Debate.

Du et al. [7] and Liang et al. [8] explore multi-agent debate for reasoning. Chen et al. [9] found that model diversity among agents was critical for performance gains in their ReConcile framework. The present work shares the emphasis on diversity but differs along three axes. First, debate is always-on; graduated dissent is conditional, triggered only when disagreement exceeds a calibrated threshold.

 doi:10.20944/preprints202603.1830.v1

This avoids the uniform cost of multi-round discussion on queries where proposers already agree. Second, debate is positional (each agent argues for its own answer); graduated dissent uses asymmetric steelmanning, requiring each proposer to reconstruct the *opposing* argument in its strongest form before critiquing it. Third, debate seeks consensus as the terminal state; graduated dissent seeks information about whether further investment in resolution is worth paying for, and explicitly preserves unresolved disagreement for human review when verification is unavailable or inconclusive.

Multi-Agent Frameworks.

AutoGen [15], CAMEL [16], and MetaGPT [17] provide general multi-agent coordination. The present contribution is orthogonal: a disagreement resolution protocol that could be implemented within any such framework.

Process Supervision.

Lightman et al. [10] showed that per-step supervision outperforms outcome supervision. This aligns with the present analysis: per-step external feedback breaks the model's single correlated context. Process supervision is an instance of external selection applied during training; the present protocol applies similar principles at inference time.

Adversarial Collaboration.

Mellers et al. [12] developed structured frameworks for expert disagreement. Kahneman [13] advocated adversarial collaboration as a methodology for resolving scientific disputes. The steelman exchange described here adapts this principle for multi-model inference.

Test-Time Scaling.

Zeng et al. [14] report that additional reasoning steps in o1-like models do not guarantee improvement due to limited self-correction. This is consistent with the information-theoretic perspective in Theorem 2: additional correlated evaluations may yield diminishing returns. The protocol proposed here offers a complementary scaling strategy: investing additional compute in decorrelated evaluation where disagreement signals warrant it.

10. Discussion

10.1. Relation to Chain-of-Thought

This paper is not an argument against chain-of-thought as such. Chain-of-thought remains valuable as a proposal mechanism. The suggestion is that chain-of-thought, especially in a single accumulating context, can be further strengthened for technical tasks when augmented with decorrelated evaluation. The protocol positions chain-of-thought within a broader inference framework: one ingredient alongside decorrelated critique and selective verification.

10.2. Relation to Test-Time Compute Scaling

Recent work on scaling test-time compute has demonstrated substantial gains from extended reasoning. An interesting question is how to allocate that additional compute most effectively. The information-theoretic framework identifies a regime where returns from additional steps within a single context may diminish: when additional steps share the same blind spots, they may consolidate confidence without adding proportional independent evidence (Theorem 2). The present approach offers a complementary scaling strategy: allocating additional compute specifically to decorrelated evaluation where disagreement signals indicate high expected information gain. This complements within-context reasoning by investing in breaking error correlation where it matters most.

The protocol also offers a structural efficiency advantage over equivalently deep single-context chain-of-thought. Clean-context evaluations operate on shorter inputs (the candidate output without the full reasoning trajectory), so each evaluation consumes less context budget than an equivalently thorough in-context critique. Additionally, proposal generation and heuristic evaluations can execute

 doi:10.20944/preprints202603.1830.v1

in parallel rather than sequentially. The total token cost may exceed a single-pass chain-of-thought, but the wall-clock latency to reach an equivalently scrutinized conclusion can be lower, because parallel decorrelated evaluations proceed simultaneously rather than waiting for a single long chain to complete. The goal is not to reduce per-token cost but to improve the reliability achieved per unit of latency, making each token of compute contribute more independent evidence about correctness.

10.3. Human Adjudication

Human judgment remains the final external criterion. The system improves triage and concentrates attention on claims surviving progressively stronger scrutiny. It does not eliminate responsibility. It makes responsibility more tractable.

The architecture explicitly preserves disagreement information for human review. When the system reports a Level 1 flagged synthesis, the human operator can examine the disagreement and decide whether to accept or investigate further. When the system reports an unresolved Level 2 disagreement after verification, the operator has full provenance: both proposals, both steelmans, both critiques, and the verification result. The goal is to ensure that human attention is allocated to the cases where it has the highest marginal value.

11. Conclusion

We introduced graduated dissent, an inference architecture for reducing correlated error in multi-model technical reasoning. The architecture is motivated by information-theoretic arguments [1] suggesting that self-evaluation may provide limited signal when generator and evaluator share failure modes, and that repeated self-critique may yield diminishing returns under strong error coupling.

The protocol addresses this by treating disagreement resolution as a resource allocation problem. Independent proposers generate candidate analyses in separated contexts. A comparator triages divergence against domain-calibrated noise floors. Only high-signal disagreements trigger expensive decorrelated evaluation: steelman exchange, adversarial cross-examination, or external verification. This design concentrates verification compute where the expected information gain is highest, complementing both single-context self-evaluation and always-on multi-agent debate.

The central hypothesis is that reliability in technical reasoning may benefit when proposal and evaluation are partially decoupled, disagreement is triaged rather than suppressed, and expensive verification is invoked selectively when dissent appears structurally meaningful. The architecture does not promise autonomous truth discovery. Its contribution is procedural: a structured method for improving what reaches human judgment, motivated by principled analysis of when and why evaluation provides information.

The theoretical framework generates testable predictions: the protocol should show its largest gains over single-context self-evaluation on tasks with high error correlation, long dependency chains, and available external verification criteria. The benchmark specifications published in this preprint constitute a pre-specified evaluation protocol for these predictions; this version will be updated with empirical results, and any deviations from the pre-specified methodology will be documented.

The principle is simple. Spend additional compute on decorrelating evaluation, not merely extending reasoning. That is the distinction that matters.

Acknowledgments: The author acknowledges the work of Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai on the Prover-Agent framework [5], which informed the verification integration design. The author thanks those who provided feedback on earlier versions of [1], which improved the theoretical foundations on which this work builds.

Use of AI Tools: AI tools assisted with drafting and editing. All ideas, methodology, architecture design, and technical content are the author's own work.

Conflicts of Interest: The author declares no conflicts of interest.

 doi:10.20944/preprints202603.1830.v1

References

1. Andrew Michael Brilliant. Limits of self-correction in LLMs: An information-theoretic analysis of correlated errors. Preprints.org, doi:10.20944/preprints202601.0892.v3, 2026.
2. Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.
3. Ken Tsui. Self-correction bench: Uncovering and addressing the self-correction blind spot in large language models. arXiv preprint arXiv:2507.02778, 2025.
4. Elliot Kim, Avi Garg, Kenny Peng, and Nikhil Garg. Correlated errors in large language models. *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.
5. Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. Prover agent: An agent-based framework for formal mathematical proofs. arXiv preprint arXiv:2506.19923, 2025.
6. Xuezhi Wang et al. Self-consistency improves chain of thought reasoning in language models. *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023.
7. Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.
8. Tian Liang et al. Encouraging divergent thinking in large language models through multi-agent debate. arXiv preprint arXiv:2305.19118, 2023.
9. Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
10. Hunter Lightman et al. Let's verify step by step. arXiv preprint arXiv:2305.20050, 2023.
11. Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.
12. Barbara Mellers, Ralph Hertwig, and Daniel Kahneman. Do frequency representations eliminate conjunction effects? An exercise in adversarial collaboration. *Psychological Science*, 12(4):269–275, 2001.
13. Daniel Kahneman. A perspective on judgment and choice: Mapping bounded rationality. *American Psychologist*, 58(9):697–720, 2003.
14. Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4651–4665, 2025.
15. Qingyun Wu et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. arXiv preprint arXiv:2308.08155, 2023.
16. Guohao Li et al. CAMEL: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36, 2023.
17. Sirui Hong et al. MetaGPT: Meta programming for a multi-agent collaborative framework. arXiv preprint arXiv:2308.00352, 2023.