PoTRE: Scaling Test-Time Reasoning via Cognitive Heterogeneity

Anonymous authors
Paper under double-blind review

Abstract

While Large Language Models (LLMs) excel at many tasks, they frequently struggle with complex reasoning that requires long-horizon planning and iterative error correction. Furthermore, standard single-stream prompting proves brittle when models encounter novel abstractions or rigorous domain constraints. We introduce PoTRE (**Po**ly-**T**opological **R**easoning **E**nsembles), a heterogeneous framework that decouples inference into four orthogonal agents: (1) Adversarial Refinement Agent, (2) Hierarchical strategic Planning Agent, (3) Spectrum Search Agent, and (4) Direct Chain Agent. A final Task-Adaptive Aggregation Layer dynamically reconciles these perspectives—via instinct driven selection, semantic synthesis, or neuro-symbolic verification—to produce a robust global solution. We evaluate PoTRE on three frontier benchmarks: ARC-AGI-2, Humanity's Last Exam (HLE), and PRBench Finance. PoTRE achieves state-of-the-art accuracy of 49.92% on HLE, surpassing the previous best official score. We demonstrate that architectural heterogeneity is an effective scaling path.

1 Introduction

Scaling Large Language Models (LLMs) has yielded remarkable capabilities across general-purpose tasks, driven largely by the optimization of next-token prediction on massive corpora OpenAI (2024); DeepMind (2025). However, a persistent "reasoning gap" remains between fluent text generation and robust, multi-step problem solving. While test-time reasoning strategies like Chain of thought prompting Wei et al. (2022) and self-consistency Wang et al. (2023) have enhanced the performance of frontier models, they remain fundamentally brittle when facing tasks that require long-horizon planning, rigorous error correction, or novel abstraction—capabilities often categorized as "System 2" thinking Kahneman (2011).

This brittleness is most evident in the emergence of next-generation benchmarks designed explicitly to break simple heuristic pattern matching. *Humanity's Last Exam* (HLE) Hendrycks et al. (2025), for instance, poses multidisciplinary problems at the frontier of human knowledge where retrieval alone is insufficient. Similarly, *ARC-AGI-2* Chollet & the ARC Prize Team (2025) tests fluid intelligence through novel visual abstractions that cannot be solved via memorized recipes, and *PRBench* Research (2025) demands high-stakes professional judgment in finance and law. On these rigorous testbeds, standard single-stream prompting—and even homogeneous ensembles—often fail. We observe that such systems frequently suffer from cognitive mode collapse: because homogeneous agents share identical probabilistic priors, their errors are highly correlated. Consequently, simply scaling the ensemble size leads to "groupthink", where agents reinforce plausible but incorrect fallacies rather than correcting them via orthogonal reasoning strategies.

Current research attempts to mitigate this by structuring reasoning into trees or graphs Yao et al. (2023); Besta et al. (2024). However, these methods typically rely on a single cognitive mode—usually the model's default generation style—replicated multiple times. We argue that robust improvements can come from ***cognitive heterogeneity***, defined here as the integration of reasoning topologies that exhibit error orthogonality. Unlike homogeneous ensembles where errors are highly correlated, orthogonal agents possess distinct distributional failure modes. We observe this empirically in our ARC-AGI-2 analysis: the Spectrum search agent, despite lower average accuracy, acts as a "Specialist" that uniquely solves 14% of tasks missed by

all other agents. Just as human decision-making benefits from the interplay between creative divergence, critical dialectic, and structured planning, LLM reasoning should decouple these processes into specialized agents that fail *differently* to succeed *collectively*.

We introduce **PoTRE** (Poly topological Reasoning Ensembles), a framework that synthesizes four such orthogonal reasoning agents to solve complex tasks. Unlike ensemble methods that simply average outputs from identical agents, PoTRE orchestrates four specialized agents: (1) a *Adversarial Refinement Agent* for debate, (2) a *Hierarchical strategic planning agent* for sub-goal decomposition, (3) a *Spectrum search agent* for breadth-first candidate search, and (4) a *Direct Chain Agent*. A final *Synthesis Agent* aggregates these candidate answers via task adaptive aggregation —whether through instinct or neuro-symbolic verification—PoTRE filters agent-specific hallucinations to construct a robust global solution.

This architecture allows us to explicitly contrast PoTRE's *parallel breadth* with "cognitive heterogeneity" against the *sequential depth* optimized by current SOTA systems. Deep Research systems often rely on long-context centralization Cai et al. (2026) or iterative homogeneous refinement Tang et al. (2026), and require frontier-model backbones (e.g., Gemini-3-Pro-Preview) to function. Our results on the HLE Open-Book benchmark with Gemini-3-Flash-Preview (55.28% vs. 51.7%/52.2%) demonstrate that parallelizing reasoning across diverse agents is a highly effective method for decoupling performance from parameter scale, proving to be a superior architectural strategy for maximizing the reasoning potential of lightweight models.

We evaluate PoTRE on three challenging reasoning benchmarks available. As shown in Figure 1, our results reveal a striking lift by scaffolding: the smaller, efficient Gemini-3-Flash-Preview model, when augmented with PoTRE, consistently outperforms the standalone Gemini-3-Pro-Preview baseline. On Humanity's Last Exam, PoTRE achieves a state-of-the-art accuracy of 49.92%. On ARC-AGI-2, our method scores 38.30%, nearly doubling the baseline performance. Finally, to demonstrate PoTRE's utility in high-stakes, real-world applications, we establish a new state-of-the-art on the domain-specific PRBench Finance Hard with an average clipped score of 0.5196. These findings highlight the ability of our cognitively heterogeneous framework to synthesize complex, conflicting information into accurate insights, demonstrating that the key to next-generation reasoning lies not merely in scaling model parameters, but in scaling inference-time cognitive diversity.

2 Related Work

Standard approaches to LLM reasoning rely on prompt engineering to induce intermediate computation. Chain of thought Wei et al. (2022) demonstrated that prompting models to emit reasoning steps significantly improves performance. Subsequent works extended this via Self-Consistency Wang et al. (2023), which marginalizes over multiple reasoning paths to filter stochastic errors.

Recent methods have moved beyond linear chains toward non-linear topologies. Tree of Thoughts Yao et al. (2023) and Graph of Thoughts Besta et al. (2024) allow models to explore search trees and aggregate thoughts. However, we argue that these methods suffer from Homogeneous Failure Modes: because they replicate the same underlying cognitive process across all branches, they are vulnerable to correlated errors. If the model possesses a specific blind spot, it generates consistent but incorrect reasoning across the entire topology.

Recent advances in "Deep Research" have shifted focus from single-turn QA to long-horizon autonomous information seeking. Current approaches largely fall into two architectural paradigms: Sequential Depth and Homogeneous Refinement. Systems like Yunque DeepResearch Cai et al. (2026) and Tongyi DeepResearch Team et al. (2025) rely on centralized, compute-heavy planning. Yunque employs a hierarchical "Manager-Worker" topology with dynamic context folding to handle long horizons, while Tongyi utilizes end-to-end reinforcement learning to optimize tool-use trajectories.

Parallel efforts focus on improving reasoning via iterative self-correction. ReThinker Tang et al. (2026) introduces a "Solver-Critic-Selector" loop that permutes confidence scores to filter hallucinations. Similarly, SETS Chen et al. (2025a) unifies "Sampling, Self-Verification, and Self-Correction" into a framework that

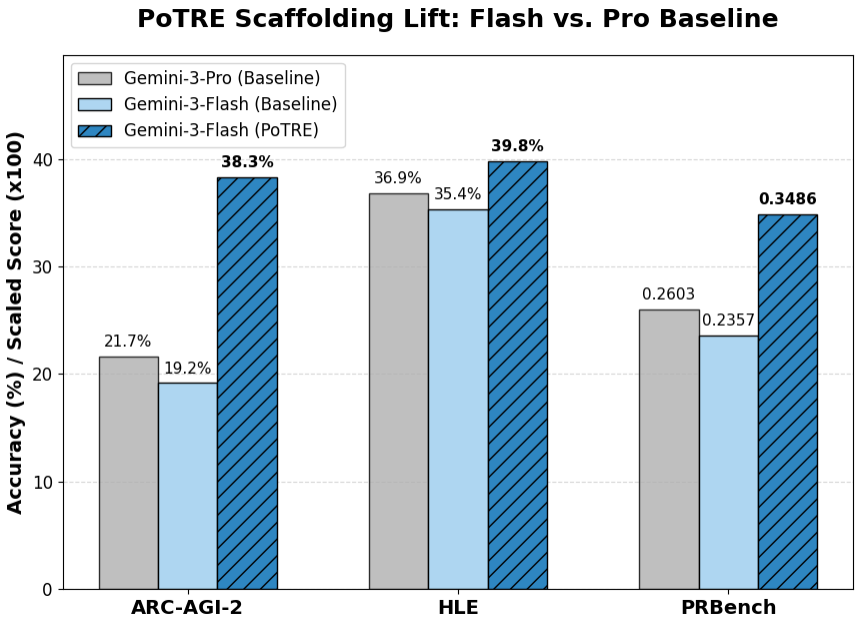

Figure 1: Comparison of the smaller Gemini-3-Flash-Preview model equipped with PoTRE (Blue) against the significantly larger Gemini-3-Pro-Preview baseline (Grey) across three diverse benchmarks. In all cases, the architectural scaffolding of PoTRE enables the efficient model to outperform the frontier model operating with prior work prompts, demonstrating that inference-time reasoning structure can effectively substitute for parameter scale.

outperforms simple voting. However, both rely on homogeneous sampling—refining a single model's outputs rather than diversifying the cognitive process itself.

In the multi-agent domain, MonoScale Shao et al. (2026) proposes a router-based framework to delegate subtasks, optimizing for "monotonic improvement" (performance stability) within an agent pool. TUMIX Chen et al. (2025b) extends this by mixing agents with distinct tool-use strategies. Yet, these approaches generally optimize a single reasoning topology. They risk cognitive mode collapse when the underlying model shares identical blind spots, as they do not simultaneously exploit orthogonal cognitive structures (e.g., combining adversarial debate with hierarchical planning) to cancel out the inherent biases of any single method.

Rather than mitigating these biases through diverse inference topologies, other recent advancements have attempted to solve the multi-agent orchestration problem by training dedicated routing models. A notable example is ToolOrchestra Su et al. (2025), which introduces a lightweight orchestrator model explicitly trained via Reinforcement Learning to dynamically route complex queries to specialized tools and frontier models. While ToolOrchestra demonstrates impressive gains on challenging benchmarks, it requires a highly specialized training pipeline, extensive data synthesis, and custom reward modeling to align the orchestrator's behavior.

Parallel to these orchestration efforts, significant progress in model reasoning has been driven by data-centric training strategies. Notably, OpenThoughts Guha et al. (2025) demonstrates that highly capable reasoning models can be developed through rigorous Supervised Fine-Tuning (SFT) data recipes. By systematically optimizing question sourcing, multi-answer sampling, and reasoning trace distillation from teacher models, OpenThoughts produced competitive open-source models. However, data-centric approaches fundamentally require expensive curation pipelines and permanent weight modifications to the underlying base model, posing practical difficulties.

In contrast to these approaches, PoTRE introduces Cognitive Heterogeneity. Instead of optimizing for sequential depth or homogeneous refinement, PoTRE enforces topological diversity by running four orthog-

onal reasoning agents in parallel. Our results demonstrate that this "width-over-depth" approach allows lightweight models to outperform the heavy-compute baselines of the sequential paradigm, effectively decoupling reasoning performance from parameter scale.

3 Methodology

We introduce PoTRE (Poly-Topological Reasoning Ensembles), a heterogeneous multi-agent framework designed to solve complex reasoning tasks by decoupling cognitive strategies into parallel reasoning agents. Unlike homogeneous ensembles that aggregate identical agents, PoTRE employs four distinct reasoning agents—Adversarial Refinement Agent via debate, Hierarchical Strategic Planning Agent, Spectrum Search Agent, Direct Chain Agent—to maximize the diversity of the solution space.

Given an input problem P, the system distributes the task in parallel to four independent sub-agents. The resulting set of intermediate solutions $\mathcal{S} = \{s_{debate}, s_{planning}, s_{spectrum}, s_{direct}\}$ is then aggregated by a final Synthesis agent to produce the optimal solution $\hat{y}$.

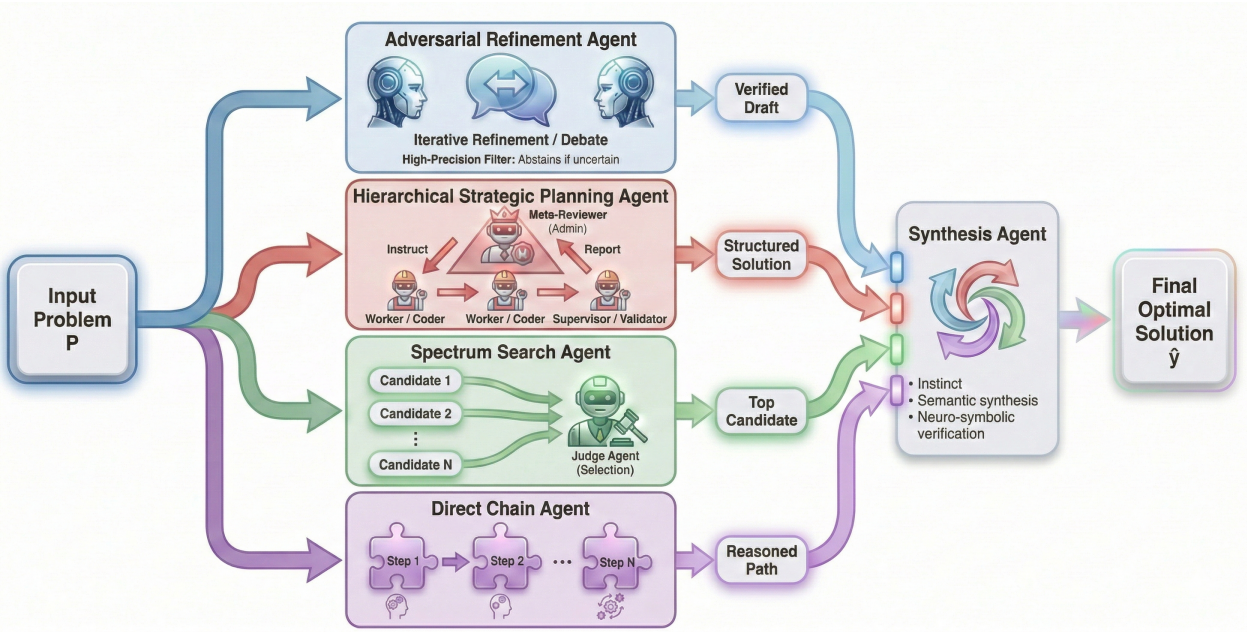

Figure 2: **Poly-Topological Reasoning Ensembles**. This diagram illustrates the proposed multi-agent reasoning process. An input problem is processed in parallel by four distinct agent-based strategies: (1) Adversarial Refinement, involving iterative debate for high-precision output, (2) Hierarchical Strategic Planning, using a nested team structure for structured solutions, (3) Spectrum Search, generating and selecting top candidate via a judge agent, and (4) Direct Chain for standard sequential reasoning. The outputs from these diverse reasoning paths (verified draft, structured solution, top candidate, and reasoned path) are aggregated by a centralized Synthesis Agent, which utilizes instinct, semantic synthesis, and neuro-symbolic verification to generate the final optimal solution.

3.1 The Orthogonal Agents

Each agent is designed to target specific failure modes. The parallel orchestration and subsequent synthesis of these agents are detailed in Algorithm 1 and algorithms for each sub-agent is given in Appendix C

Algorithm 1 PoTRE: Core Orchestration and Task-Adaptive Synthesis

Require: Input query Q, Task Type $\mathcal{T}_{type} \in \{\text{Constrained}, \text{Open-Ended}, \text{Rule-Based}\}$, Model $\mathcal{M}$
Ensure: Final synthesized answer A^*

% Phase 1: Parallel Agent Triggering (Independent Execution)
$s_{direct} \leftarrow \text{Direct_Chain_Agent}(Q, \mathcal{M})$ ▷ Sequential step-by-step logic
$s_{spectrum} \leftarrow \text{Spectrum_Search_Agent}(Q, \mathcal{M})$ ▷ Solution diversity
$s_{debate} \leftarrow \text{Adversarial_Refinement_Agent}(Q, \mathcal{M})$ ▷ Adversarial refinement
$s_{planning} \leftarrow \text{Hierarchical_Strategic_Planning_Agent}(Q, \mathcal{M})$ ▷ Planner → Executor → Verifier

% Phase 2: Output Collection
$\mathcal{S} \leftarrow \{s_{direct}, s_{spectrum}, s_{debate}, s_{planning}\}$

% Phase 3: Synthesis Agent (Task-Adaptive Aggregation)
if $\mathcal{T}_{type}$ == Constrained **then**
  $A^* \leftarrow \text{Instinct_Selection}(Q, \mathcal{S}, \mathcal{M})$ ▷ Instinct based selection
else if $\mathcal{T}_{type}$ == Open-Ended **then**
  $A^* \leftarrow \text{Qualitative_Synthesis}(Q, \mathcal{S}, \mathcal{M})$ ▷ Semantic Synthesis
else if $\mathcal{T}_{type}$ == Rule-Based **then**
  $A^* \leftarrow \text{Neuro_Symbolic_Verifier}(Q, \mathcal{S}, \mathcal{M})$ ▷ Logic-Consistency validation
end if
return A^*

3.1.1 Adversarial Refinement Agent

This agent addresses hallucination through iterative adversarial refinement. Two agents—a *Proposer* and a *Verifier*—engage in a multi-turn dialogue (up to $T = 5$ turns) to scrutinize and iteratively improve the solution quality. Rather than merely filtering outputs, the Verifier provides constructive critique that guides the Proposer toward a more robust answer.

The process follows a strict consensus:

- **Proposal & Critique:** The Proposer generates an initial candidate (rationale and answer). The Verifier acts as a rigorous critic, scrutinizing the output for logical gaps, factual inconsistencies, or constraint violations.
- **Iterative Refinement:** If the Verifier rejects the candidate, it provides specific feedback. The Proposer must then generate a revised candidate that addresses these critiques.
- **Final Consensus:** The answer is only accepted if the Verifier explicitly outputs a `STATUS: APPROVED` token.

Selective Abstention: This agents operate on a strict "Pass/Fail" basis. If the Verifier remains unsatisfied after the maximum turns, the agent abstains (returning $\emptyset$). This agent ensures that only rigorously verified solutions are propagated to the Synthesizer, preventing the contamination of the final decision process with uncertain outputs.

3.1.2 Hierarchical Strategic Planning Agent

This agent implements a Hierarchical cognitive architecture designed for long-horizon planning and strategic course correction. Operating with a maximum context budget of $T_{max} = 10$ turns, the system functions as a specialized role-based team:

- **Planner:** Acts as the strategic core that structures the reasoning process prior to execution. Depending on the input modality, it either decomposes complex queries into a sequence of 2–3 logical sub-goals or abstracts a single universal transformation rule. This component enforces dynamic

derivation, prioritizing the identification of global patterns (such as visual legends or alignment keys) over case-specific hard-coding.

- **Executor:** Implements the proposed solution, producing executable traces or detailed logical derivations.
- **Verifier:** Evaluates the execution against the Planner's proposal, strictly enforcing constraint satisfaction.
- **Overseer:** Acts as a dynamic control agent: if the worker agents get stuck repeating the same error or fail to make progress, the Overseer intervenes to halt the current path and force the Planner to generate a fresh strategy.

Active Meta-cognitive Control Loop. To mitigate the diminishing returns of unguided self-correction, we employ a agent of autonomous strategic pivoting. The "overseer" performs continuous state-space monitoring on the dialogue history:

1. **Trajectory Analysis:** The system scans for "Strategic Failures," such as repeated hypothesis cycling, logical circularity, or recurring execution errors.
2. **Diagnosis:** Upon detecting a failure mode, the system isolates the root cause—distinguishing between operational faults (e.g., calculation or syntax errors) and strategic misalignment (e.g., getting stuck in a local optimum).
3. **Strategic Intervention (The Pivot):** Upon diagnosing a strategic stall, the overseer executes a two-stage intervention:
   - *Hypothesis Rejection:* It explicitly invalidates the current reasoning path (e.g., "The assumption that color dictates movement is leading to contradictions; abandon this premise").
   - *Directive Synthesis:* Instead of selecting from a pre-defined list of error codes, the model dynamically generates a high-level natural language instruction tailored to the specific failure context (e.g., "Re-analyze the grid using connectivity-based object detection instead of pixel-wise color matching").

   This agent forces a hard transition from passive debugging (fixing code) to active re-planning (changing logic), effectively breaking the agent out of recursive loops without manual intervention.

3.1.3 Spectrum Search Agent

To address the cascading logical errors prevalent in single-trajectory generation, we employ a parallelized ensemble approach as a form of Test-Time Scaling Snell et al. (2024). This agent prioritizes breadth of exploration over depth of planning:

Divergent Candidate Generation. To ensure diverse solution discovery, the system spawns a parallel batch of N independent worker threads. Each worker generates a unique candidate solution using a non-zero temperature. This stochastic sampling allows the system to escape local optima that typically trap deterministic greedy decoding strategies Wang et al. (2023).

Task-Specific Candidate Selection. To select a single final answer ($s_{spectrum}$) from the N independent agents, we apply a filtering strategy that adapts to the nature of the specific task:

- **Constraint-Guided Filtering:** For tasks that feature formal test cases or verifiable rules, we leverage the given premises as a "weak verifier" to perform a multi-stage refinement process:
  1. *Hypothesis Verification:* First, the reasoning pipeline enforces a rigorous negative constraint check. Before finalizing a candidate solution, the agent must verify that its proposed logic holds conceptually true against all provided problems and boundary conditions. This internal critique loop filters out logical inconsistencies early, ensuring only robust, fully validated hypotheses proceed to the execution phase.

2. *Execution Pruning:* When applicable, the validated hypotheses are translated into an executable format (e.g., programmatic code or formal logic) and evaluated against the known constraints. Any candidate that fails to strictly reproduce the expected target conditions is pruned as functionally invalid.
3. *Consensus Selection:* From the surviving subset of verified candidates, we select the final answer via majority voting. This funnel ensures that consensus is built only upon candidates that have passed both conceptual (rule-based) and functional (execution-based) verification.

- **Semantic Judging:** For open-ended reasoning where no execution environment exists, we rely on a LLM as a Judge. This agent evaluates candidates based on internal logical consistency and adherence to problem constraints, selecting the single most robust solution without external execution signals.

3.1.4 Direct Chain Agent

To ensure the system remains grounded in direct reasoning, we retain a standard Chain of thought. This agent produces a solution s_{direct} using step-by-step reasoning, serving as a baseline anchor for the more complex agentic systems. We employ a task-adaptive prompting strategy: Zero-Shot CoT Kojima et al. (2023) for open-ended reasoning to avoid biasing the model with potentially irrelevant examples, and Few-Shot CoT for inductive tasks where input-output demonstrations are essential for defining the problem space.

3.2 Synthesis Agent: Task-Adaptive Aggregation

The final component of PoTRE is the Synthesis Agent. Recognizing that different problem structures require distinct evaluation strategies, we employ a task-adaptive approach that selects the optimal synthesis method based on the expected output modality:

- **Instinct-Driven Selection for Constrained Outputs:** For tasks requiring highly specific or constrained responses (e.g., short-answer or multiple-choice formats), we intentionally transition away from rigid majority voting. Instead, we utilize an instinct based prompting strategy. The Synthesis Agent relies solely on its instinct to assess the candidate set $\mathcal{S}$, rather than merely counting exact textual matches, the agent identifies the most robust and well-supported conclusion, effectively filtering out consensus-driven hallucinations.
- **Qualitative Synthesis for Open-Ended Generation:** For complex reasoning tasks requiring long-form professional judgment, exact-match statistical voting is inherently ill-defined, as no two reasoning trajectories are identical. In these scenarios, we deploy a qualitative integration mechanism to critically evaluate the independent answers from agents, resolve conflicting viewpoints, and merge their most verifiable arguments into a single, cohesive final response.
- **Logic-Consistency Verification for Rule-Based Tasks:** For tasks rooted in formal logic, pattern induction, or strict constraint satisfaction, we employ a neuro-symbolic verifier, it deeply evaluates the generalization logic of the surviving candidates. It is prompted to verify whether the proposed solution consistently and flawlessly applies the core rules derived from the problem's examples, prioritizing candidates that demonstrate high structural and logical coherence.

4 Experimental Setup

4.1 Benchmarks

We evaluate PoTRE across three diverse benchmarks. For each dataset, we utilize official evaluation splits to ensure comparability with public leaderboards.

- **Humanity's Last Exam (HLE):** Introduced by the Center for AI Safety Hendrycks et al. (2025), HLE represents the current ceiling for closed-ended academic reasoning, filtering out contamination

by focusing on questions that require expert-level synthesis across 2,500 diverse subjects. We evaluate on the full dataset (N = 2, 500), covering domains from humanities to advanced STEM.

- **ARC-AGI-2:** Building on the original Abstraction and Reasoning Corpus, ARC-AGI-2 Chollet & the ARC Prize Team (2025) targets fluid intelligence. Unlike language-heavy tasks, ARC requires inducing programs from few-shot visual examples, a domain where standard CoT often fails due to a lack of grounded priors. We evaluate on a high-difficulty public eval set (N = 120 tasks), specifically selected to stress-test frontier capabilities on nested abstractions where standard baselines typically fail.
- **PRBench:** Scale AI's Professional Reasoning Benchmark Research (2025) evaluates performance in high-stakes domains (Finance and Law) using expert-curated rubrics rather than simple exact-match accuracy, testing a model's ability to adhere to rigorous constraints. We specifically utilize the "Hard" subset ($N = 300$ samples) to rigorously test the model's ability to apply specialized financial knowledge.

4.2 Baselines

To strictly isolate the impact of the PoTRE architecture from prompt engineering variables, we take prompts from prior work for baseline comparisons. Rather than using naive zero-shot prompts, we utilize state-of-the-art prompt templates derived from recent literature for both models Zhang et al. (2025b;a).

- **Baseline Models:** We use Gemini-3-Flash-Preview, Gemini-3-Pro-Preview and Gemini-3.1-Pro-Preview as small and large versions of Gemini 3 with prior work prompts.
- **Self-Consistency (SC) Baselines:** To distinguish the impact of reasoning topology from simple ensemble scaling, we evaluate all versions using Self-Consistency at two distinct scales ($N = 8$ and $N = 16$) Wang et al. (2023). We generate up to 16 independent trajectories per problem and aggregate results via majority voting. This serves as a rigorous sampling baseline, allowing us to strictly verify whether PoTRE's gains arise from *cognitive heterogeneity* (diverse reasoning types) rather than just *homogeneous repetition*.
- **Official Leaderboard Benchmarking:** To ensure external validity, we benchmark PoTRE against the official public leaderboards for *Humanity's Last Exam* (HLE) and the *PRBench* "Finance Hard" subset. This aligns our evaluation with the current industrial state-of-the-art, allowing us to strictly verify PoTRE's performance against the highest-performing commercial systems in a standardized, reproducible setting. Furthermore, we are not comparing ARC-AGI 2 with leaderboard because leaderboard evaluates on semi-private dataset and we are using public evaluation set.

4.3 Evaluation Protocol

To ensure robust and fair scoring, we strictly adhere to the official evaluation protocols for each benchmark:

- **ARC-AGI-2:** Evaluation is performed using standard Programmatic Validation. A solution is scored as correct only if the generated python script produces the exact pixel-perfect output grid for the test input. We do not use partial credit; the validation is strictly binary and executed in a deterministic environment.
- **Humanity's Last Exam (HLE):** Given the complexity of the dataset, we utilize a blind LLM-as-a-Judge (Gemini-3-Flash-Preview-Preview) to grade responses. The judge evaluates semantic equivalence between the predicted answer and ground truth, robustly handling variations in formatting (e.g., "50%" vs "0.5") to determine final accuracy. We utilize the official prompt for evaluation provided by the authors Hendrycks et al. (2025).
- **PRBench Finance:** To ensure strict comparability, we utilize the official evaluation script provided by the benchmark authors Research (2025), modifying only the inference backend to target our evaluated models (Gemini-3-Pro-Preview-Preview).

The scoring logic follows the formal definition in Appendix D.1 Research (2025) of the original benchmark. For a model M producing response m_j to prompt p_j, an LLM judge assigns binary indicators $I_{j,i} \in \{0, 1\}$ for each rubric criterion with weight $w_{j,i}$. The score for a single response s_j is calculated as:

$$s_j = \frac{\sum_{i=1}^{k_j} w_{j,i} I_{j,i}}{\sum_{i:w_{j,i}>0} w_{j,i}} \tag{1}$$

The final score $S(M)$ is the mean over all n prompts, clipped at zero to account for potential negative penalties:

$$S(M) = \max\left(0, \frac{1}{n}\sum_{j=1}^{n} s_j\right). \tag{2}$$

4.4 Implementation Details

To evaluate the scalability and efficiency of our framework, we implemented PoTRE using different backbone models. The architecture was orchestrated using an asynchronous parallel execution pipeline with distinct behavioral configurations for each agent:

- **Role-Conditioned Prompting:** Each agent was instantiated with a specialized system prompt defining its cognitive role. Full prompt templates for all agents are provided in Appendix A. This ensures that the multi-agent system generates orthogonal candidates through fundamental differences in problem-solving perspectives rather than mere stochasticity.
- **Modality-Specific Execution:** The framework adapts its output format to the intrinsic nature of each benchmark. For HLE and PRBench, agents generate natural language reasoning terminating in a final text option. For ARC-AGI-2, we deployed the Neuro-Symbolic interface: all four agents were instructed to generate python code. These scripts were executed in a secured sandboxed environment, with execution results (and errors) fed back to the agents, ensuring candidates were grounded in algorithmic logic rather than visual hallucination.
- **Operational Constraints:** To prevent unconstrained generation, we enforced strict turn-based limits tailored to each agent's depth: *Adversarial Refinement Agent* was capped at $T = 5$ turns (to force rapid consensus), while the *Hierarchical Strategic Planning Agent* was allocated $T = 10$ turns (to allow for long-horizon planning), *Spectrum Search Agent* is set to generate N=8 candidates and *Direct Chain Agent* is capped to generate single answer only.
- **Adaptive Aggregation:** The final Synthesis agent receives the original problem context P and the set of candidate solutions $\mathcal{S}$. It applies a task-specific protocol to resolve conflicts: for open-ended tasks (e.g., PRBench), it integrates the strongest verifiable arguments into a cohesive response; for constrained tasks (e.g., HLE), it is prompted to "trust its instinct," relying on reasoned intuition rather than statistical consensus; and for spatial reasoning (e.g., ARC), it employs neuro-symbolic verification to confirm logical consistency. This ensures the global solution is strictly grounded in the most robust signal available—whether that is semantic coherence, intuitive logic, or formal constraint satisfaction.

5 Results

We evaluate PoTRE across three benchmarks, comparing performance against standard single-agent baselines and Self-Consistency ensembles at two scales ($N = 8$ and $N = 16$) using Gemini-3-Pro-Preview, Gemini-3-Flash-Preview, and Gemini-3.1-Pro-Preview The results, summarized in Table 1, demonstrate that our heterogeneous multi-agent architecture generally yields superior returns compared to simply scaling homogeneous samples.

Table 1: Performance comparison of PoTRE against established baseline prompts from prior work and Self-Consistency (SC) with 8 and 16 samples. Evaluation metrics comprise *Accuracy* for HLE and ARC-AGI 2, and *Average Clipped Score* for PRBench. All reported results denote Pass@1 performance. Key Insight: PoTRE demonstrates consistent improvements over SC baselines across most settings (with one exception for the Pro model on ARC-AGI 2). Notably, our four-agent scaffolding provides substantial lift(†), enabling the PoTRE-augmented Flash models to outperform the un-augmented Pro baselines.

		Baseline Evaluation			PoTRE Evaluation	
Benchmark	Model	Prior Work Prompts	SC ($N = 8$)	SC ($N = 16$)	Score	Improv.‡
Humanity's Last Exam	Gemini-3-Flash-Preview	35.36%	39.12%	37.32%	**39.80%**†	+4.44
	Gemini-3-Pro-Preview	36.88%	39.00%	40.20%	**44.00%**	+7.12
	Gemini-3.1-Pro-Preview	42.15%	46.15%	45.95%	**49.92%**	**+7.77**
ARC-AGI 2	Gemini-3-Flash-Preview	19.16%*	33.33%	36.67%	**38.33%**†	**+19.17**
	Gemini-3-Pro-Preview	21.66%*	22.50%	28.33%	**33.33%**	+11.67
	Gemini-3.1-Pro-Preview	59.16%	75.00%	**80.00%**	78.33%	**+19.17**
PRBench Finance Hard	Gemini-3-Flash-Preview	0.2357	0.1869	0.1888	**0.3486**†	+0.1129
	Gemini-3-Pro-Preview	0.2603	0.2352	0.2427	**0.3319**	+0.0716
	Gemini-3.1-Pro-Preview	0.2713	0.2404	0.2421	**0.3931**	**+0.1218**

†Indicates Scaffolding Lift (Flash PoTRE > Gemini-3-Pro-Preview Baseline).

‡Improvement calculated vs. Baseline (Prior work prompts).

*Prior work prompts degraded performance on ARC-AGI; standard CoT performs better (see Component Analysis).

Notably, PoTRE outperforms the computationally expensive Self-Consistency ($N = 16$) baseline in all except one experimental settings. Crucially, we observe that simply scaling the homogeneous ensemble size often resulted in performance degradation. On PRBench, all models performance drops from SC (N=8) –> SC (N=16), validating that "more of the same" cannot solve reasoning blind spots; in contrast, PoTRE's architectural heterogeneity provides stability, ensuring that additional compute yields strictly positive returns. Exception to this trend is observed in the Gemini-3.1-Pro-Preview model on ARC-AGI-2, where the $N = 16$ ensemble (80.00%) marginally surpasses the PoTRE system (78.33%).

5.1 Performance Analysis

State-of-the-Art on HLE On *Humanity's Last Exam* (HLE), PoTRE utilizing the Gemini-3.1-Pro-Preview backbone achieved an accuracy of **49.92%**, surpassing the baseline of 42.15% by a margin of **7.77** percentage points. These results represents a new state-of-the-art on the official leaderboards.[1]. Even with the smaller Flash model augmented with PoTRE reached 39.80%, outperforming the standard Gemini-3-Pro-Preview baseline.

Scaffolding Lift on ARC-AGI A striking finding is observed on the ARC-AGI 2 benchmark. Although the Gemini-3-Pro-Preview baseline (21.66%) initially outperformed the standalone Gemini-3-Flash-Preview model (19.16%), PoTRE effectively reversed this. When augmented with PoTRE, the Flash model's performance jumped to 38.33%, a massive +19.17 percentage points improvement over its baseline. Crucially, this result outperforms even the larger $N = 16$ self-consistency ensemble (36.67%), confirming that architectural diversity yields significantly higher returns than simple sampling scale.

5.2 Ablation Study: Contribution of Individual agents

To isolate the source of PoTRE's performance gains, we analyzed the standalone accuracy of each agent before generating the final answer.

[1]Official Leaderboard: `https://scale.com/leaderboard/humanitys_last_exam` and `https://agi.safe.ai/`

Table 2: **Component Analysis on HLE with $N = 2500$ samples.** The "Oracle" row represents the theoretical upper bound where at least one agent produced the correct answer. Key Insight: We observe a divergence in optimal reasoning topology: the Pro models (Gemini-3-Pro-Preview and 3.1-Pro) benefit most from the spectrum search agent, while Gemini-3-Flash-Preview achieve their highest accuracy via the Adversarial Refinement Agent. Furthermore, PoTRE Final Synthesis successfully and consistently outperforms the best individual reasoning agent across all evaluated models, though a gap remains compared to the Oracle upper bound.

	Gemini-3-Flash-Preview		Gemini-3-Pro-Preview		Gemini-3.1-Pro-Preview	
Sub-Agents	Solved	Accuracy	Solved	Accuracy	Solved	Accuracy
Adversarial Refinement Agent	**948**	**37.92%**	978	39.12%	1175	47.00%
Hierarchical strategic planning agent	883	35.32%	976	39.04%	1134	45.36%
Spectrum search agent	921	36.84%	**1048**	**41.92%**	**1210**	**48.40%**
Direct chain agent	870	34.80%	949	37.96%	1018	40.72%
Oracle Upper Bound	**1199**	**47.96%**	**1269**	**50.76%**	**1399**	**55.96%**
PoTRE Final Synthesis	995	39.80%	1100	44.00%	1248	49.92%

5.2.1 Humanity's Last Exam (HLE)

Table 2 presents the component breakdown for HLE ($N = 2500$). The results reveal a significant "Oracle Gap": while the PoTRE agents collectively generated correct solutions for more than 50% of the dataset (Oracle: 55.96%), the realized performance was constrained by selection dynamics.

Our component analysis reveals a distinct divergence in optimal reasoning topologies across different model tiers. Specifically, Gemini-3-Flash-Preview model achieve their peak single-agent performance by leveraging the Adversarial Refinement Agent (37.92%). In contrast, the more capable Gemini-3-Pro-Preview and Gemini-3.1-Pro-Preview models benefit most from the broad exploration provided by the Spectrum Search Agent (yielding 41.92% and 48.40% accuracy). Crucially, the PoTRE Final Synthesis strictly and consistently outperforms the best individual agent for every evaluated model. We attribute this systematic lift directly to our instinct-driven discriminative selection approach. By framing the final answer resolution as a zero-shot, heuristic-based choice—explicitly instructing the synthesis agent to "trust its instinct" without generating further deliberative text—the system significantly mitigates the verification bottleneck. This forces the model to rely purely on its strong discriminative priors rather than over-analyzing and rejecting valid paths. However, while PoTRE successfully drives final accuracy higher (e.g., 49.92% for Gemini-3.1-Pro-Preview), the remaining gap to the theoretical Oracle upper bound (55.96%) highlights that perfect candidate routing remains an open challenge.

5.2.2 PRBench Finance

Table 3 details our component analysis on the domain-specific PRBench Finance dataset, reinforcing the overarching trend of multi-agent superiority while revealing key differences in optimal single-agent topologies. The Gemini family exhibits varied preferences (Gemini-3-Flash-Preview peaks via the Adversarial Refinement Agent at 0.3068, while Gemini-3-Pro-Preview relies predominantly on Direct Chain Agent at 0.3266). Despite these differing baselines, the PoTRE Final Synthesis strictly outperforms the best individual reasoning agent for every evaluated model. We attribute this systematic lift to our Synthesis agent. Unlike the discriminative, instinct-based selection utilized for general QA tasks, complex financial evaluations require constructive fusion. By prompting the synthesis agent as an "Expert Finance Judge," the system dynamically constructs a comprehensive "Super-Answer"—extracting the strongest computational, analytical, and formatting elements from all sub-agents while actively performing sanity checks on conflicting data. Furthermore, this ablation clearly illustrates the lift observed by scaffolding: the smaller Gemini-3-Flash-Preview model, when equipped with the PoTRE framework (0.3486), successfully surpasses both the baseline CoT and the fully scaffolded implementation of the larger Gemini-3-Pro-Preview model (0.3319). This indicates

Table 3: **Component Analysis on PRBench Finance (Hard, $N = 300$).** Performance breakdown of individual agents compared to the final synthesis. PoTRE consistently yields the highest Average Clipped scores across all evaluated models, highlighted by agent specific gains and significant lift by scaffolding for Gemini-3-Flash-Preview where it outperforms Gemini-3-Pro-Preview in all cases.

	Gemini-3-Flash-Preview		Gemini-3-Pro-Preview		Gemini-3.1-Pro-Preview	
Sub-Agents	Avg Norm	Avg Clip	Avg Norm	Avg Clip	Avg Norm	Avg Clip
Adversarial Refinement Agent	**0.3575**	**0.3068**	0.3550	0.3041	**0.3916**	**0.3444**
Hierarchical Strategic Planning Agent	0.2839	0.2257	0.3240	0.2705	0.3666	0.3181
Spectrum Search Agent	0.2456	0.1880	0.3695	0.3198	0.3913	**0.3444**
Direct Chain Agent	0.3433	0.2919	**0.3752**	**0.3266**	0.3679	0.3176
PoTRE Final Synthesis	**0.3958**	**0.3486**	**0.3804**	**0.3319**	**0.4363**	**0.3931**

Table 4: **Component Analysis on ARC-AGI 2 ($N = 120$).** PoTRE Final Synthesis consistently elevates overall accuracy beyond the best standalone agent for every evaluated model. Furthermore, we observe the Spectrum Search Agent acting as a critical "specialist," uniquely solving hard spatial tasks that completely baffled the rest of the ensemble.

	Gemini-3-Flash-Preview		Gemini-3-Pro-Preview		Gemini-3.1-Pro-Preview	
Sub-Agents	Standalone	Exclusive	Standalone	Exclusive	Standalone	Exclusive
Adversarial Refinement Agent	3	3	25	12	31	0
Hierarchical Strategic Planning Agent	5	5	5	3	38	2
Spectrum Search Agent	**35**	**17**	18	**14**	50	8
Direct Chain Agent	28	10	**27**	8	**89**	**30**
Oracle Upper Bound	**53**	–	**56**	–	**102**	–
PoTRE Final Synthesis	**46**	–	**31**	–	**94**	–

that robust, constructive multi-agent synthesis can effectively compensate for underlying parameter scale in specialized, highly technical domains.

5.2.3 ARC-AGI 2

Table 4 details our component analysis on the ARC-AGI 2 benchmark, revealing a stark contrast in how different models navigate non-linguistic reasoning. A striking observation is the critical role of the Spectrum Search Agent as a "specialist" for generating diverse structural hypotheses. This is most pronounced with Gemini-3-Flash-Preview, which relied heavily on this divergent exploration to crack spatial tasks that baffled the rest of the ensemble, achieving 35 standalone solves with a remarkable 17 being entirely exclusive. Despite this extreme variance in single-agent capabilities, the PoTRE Final Synthesis consistently outpaces the best standalone agent for every model evaluated (e.g., boosting Gemini-3-Flash-Preview from 35 to 46, and Gemini-3.1-Pro-Preview from 89 to 94). We attribute this robust lift to the active verification mechanics embedded in our synthesis agent. Rather than employing a naive majority vote, the synthesis agent utilize a neuro-symbolic approach. By explicitly prompting the agent to verify candidate grids against the task's provided training examples, the system effectively tests and validates the inferred rules against known input-output pairs before finalizing an answer. The synthesis agent dynamically selects flawless candidates, corrects divergent errors, or derives the solution from scratch. By restricting the final output to a strict JSON grid without textual explanation, we suppress hallucinated justifications, successfully harnessing the ensemble's diverse topological strengths to drive performance higher on this notoriously resistant benchmark.

Pass@2 Performance on ARC-AGI 2: Because the official ARC-AGI leaderboard permits two submission attempts per task, we extend our evaluation to include Pass@2 performance metrics. Under this relaxed constraint, all evaluated models demonstrated notable gains. Most prominently, Gemini-3.1-Pro-Preview improved from 94 to 104 successfully solved tasks, elevating its overall accuracy from 78.33% to a state-of-the-art **86.66**% on public evaluation set. Consistent improvements were observed across the broader model family: Gemini-3-Pro-Preview increased its yield from 40 to 45 solved tasks (33.33% $\rightarrow$ 37.50%), while the highly efficient Gemini-3-Flash-Preview advanced from 46 to 54 solved tasks (38.33% $\rightarrow$ 45.00%).

5.3 Summary of Ablation Study

Across three fundamentally diverse benchmarks—Humanity's Last Exam (advanced academic reasoning), PRBench Finance (domain-specific calculation), and ARC-AGI 2 (neuro-symbolic spatial logic)—our component analysis reveals several consistent insights regarding multi-agent architectures:

1. Universal Superiority of Final Synthesis: Regardless of the underlying model tier or the specific domain, the PoTRE Final Synthesis strictly and consistently outperforms the best individual agent. By adaptively shifting the synthesis strategy—using instinct-driven discriminative selection for advanced academic reasoning (HLE), generative "Super-Answer" fusion for finance, and neuro-symbolic train-pair verification for spatial logic—the synthesis agent substantially mitigates the verification bottleneck and better leverages the diverse strengths of the ensemble. However, we note that a consistent gap remains between the Final Synthesis performance and the theoretical Oracle upper bound. This highlights that while our ensemble approach is highly effective, achieving perfect candidate selection remains an open challenge for future research.

2. Task and Model-Dependent Topologies: Our ablations demonstrate that there is no single "silver bullet" reasoning structure. Optimal single-agent performance is highly sensitive to both the task and the model's innate priors. For instance, Gemini-3-Flash-Preview relies heavily on the Adversarial Refinement Agent on PRBench Finanace whereas relies on divergent exploration of the Spectrum Search Agent to act as a crucial "specialist" on ARC-AGI 2 (yielding 17 exclusive solves). This extreme variance underscores the necessity of a multi-agent framework; a diverse ensemble ensures that at least one agent's topology aligns with the specific cognitive demands of the problem.

3. The "Scaffolding Lift" Phenomenon: Perhaps the most practically significant finding is the consistent emergence of Scaffolding Lift. On both general and specialized tasks, deploying the smaller, more efficient Gemini-3-Flash-Preview model within the PoTRE framework allows it to routinely surpass the performance of the significantly larger, unaugmented Gemini-3-Pro-Preview baseline. This indicates that robust, multi-agent structural scaffolding can effectively compensate for raw parameter scale, offering a highly efficient pathway to state-of-the-art reasoning capabilities.

5.4 Extension: Open-Book Generalization with Search

To evaluate the universality of the PoTRE framework, we extended the architecture to an "Open-Book" setting by equipping the four reasoning agents with access to Google Search. This allows for a direct comparison against frameworks like ReThinker Tang et al. (2026) and tool-heavy "Deep Research" agents like Yunque DeepResearch Cai et al. (2026) and Tongyi DeepResearch Team et al. (2025).

5.4.1 Comparison to State-of-the-Art

Table 5 presents a comprehensive comparison of PoTRE against state-of-the-art baselines on the HLE text-only benchmark ($N = 2,158$). Three key findings emerge:

- **Scaffolding Lift in Search:** In the open-book setting, the lightweight Gemini-3-Flash-Preview model (55.28%) outperforms the larger Gemini-3-Pro-Preview model (53.85%). This suggests that for tool-use tasks, the *latency-throughput advantage* of Flash (enabling wider parallel search in the Spectrum search agent) outweighs the *parameter-depth advantage* of Pro.

Table 5: **Open-Book Performance Comparison.** PoTRE establishes a new state-of-the-art on the HLE text-only subset with Gemini-3.1-Pro-Preview (60.24%). Additionally, PoTRE (Flash) demonstrates significant *Scaffolding Lift*, outperforming the heavier Yunque and ReThinker baselines despite utilizing a highly cost-efficient backbone.

System	Base Model	Mode	Accuracy
Baselines			
Tongyi DeepResearch Team et al. (2025)	Qwen-30B	Open-Book	32.90%
Yunque DeepResearch Cai et al. (2026)	Gemini-3-Pro-Preview	Open-Book	51.70%
ReThinker Tang et al. (2026)	Gemini-3-Pro-Preview	Open-Book	52.20%
PoTRE (Ours)			
PoTRE (Pro)	Gemini-3-Pro-Preview	Open-Book	53.85%
PoTRE (Flash)	Gemini-3-Flash-Preview	Open-Book	**55.28%**
PoTRE (3.1 Pro)	**Gemini-3.1-Pro-Preview**	**Open-Book**	**60.24%**

- **Outperforming SOTA:** PoTRE (Flash) achieves 55.28%, explicitly outperforming ReThinker framework (52.2%) and the Yunque DeepResearch baseline (51.7%). This demonstrates a significant "Scaffolding Lift," proving that our "Cognitive Heterogeneity" architecture enables lightweight models to surpass frontier proprietary systems.
- **Superiority over Open-Source Agents:** PoTRE significantly outperforms the Tongyi DeepResearch agent (32.9%), establishing a new state-of-the-art for open-architecture research systems on this benchmark.
- **Agent Comparison (Depth vs. Breadth):** We contrast PoTRE's *parallel breadth* (cognitive heterogeneity) against the *sequential depth* optimized by current SOTA systems. While Yunque DeepResearch relies on long-context centralization Cai et al. (2026) and ReThinker relies on iterative homogeneous refinement Tang et al. (2026), both require frontier-model backbones (Gemini-3-Pro-Preview) to function. Our results (55.28% vs. 51.7%/52.2%) demonstrate that parallelizing reasoning across diverse agents is a highly effective method for decoupling performance from parameter scale, proving to be a superior architectural strategy for maximizing the reasoning potential of lightweight models.

Full Set Verification and SOTA Comparison. To ensure these findings are robust, we also evaluated PoTRE on the full HLE validation set ($N = 2,500$), which includes multimodal and complex queries. The Scaffolding Lift remains consistent: Gemini-3-Flash-Preview achieves 53.48% (1337/2500), maintaining its lead over the standalone Gemini-3-Pro-Preview baseline (51.88%). Furthermore, Gemini-3.1-Pro-Preview achieves **58.40%**(1460/2500) maintaining its lead amongst all other models.

5.5 Impact of Search Augmentation on Sub-Agents

Figure 3 illustrates the profound impact of search augmentation across our four orthogonal reasoning agents on the HLE dataset. While external retrieval universally elevates performance, the magnitude of these gains reveals critical insights into the distinct cognitive bottlenecks of each agent structure:

- **Effect on Direct Chain Agent:** The most dramatic improvement is observed in the Direct Chain Agent. Operating under closed-book constraints, this standard reasoning topology performed the poorest (40.72%). However, upon introducing search augmentation, it experienced a massive +14.68% surge, bringing it perfectly in line with the other agents at 55.40%. This indicates that

Table 6: **MCQ Performance.** By testing on the MonoScale subset, we confirm that PoTRE (Open-Book) significantly outperforms the agentic scaling baseline (65.48% vs. 19.90%). This demonstrates that *reasoning architecture* yields far higher gains than merely scaling the number of agents ($N = 10$).

Method	Model	MCQ Accuracy
MonoScale ($N = 10$ Agents)	Qwen-3	19.90%
PoTRE (Ours)	Gemini-3-Flash-Preview	**60.74**%
PoTRE (Ours)	Gemini-3-Pro-Preview	60.07%
PoTRE (Ours)	Gemini-3.1-Pro-Preview	**65.48%**

traditional Chain-of-Thought reasoning is disproportionately bottle-necked by parametric memory limits. When forced to rely solely on internal weights for highly specialized HLE queries, the agent confidently hallucinates. Search acts as a massive equalizer, substituting flawed internal retrieval with grounded external facts, allowing the model's pure logical sequencing to shine.

- **Amplifying Exploration:** The Spectrum Search Agent achieved the highest absolute search-augmented accuracy (57.20%), building upon an already strong closed-book foundation (48.40%). Because this agent's core topology is designed for breadth-first exploration and trying different things across diverse hypotheses, injecting real-time search data increases the quality of its candidate generation. Search gives the spectrum search agent a substantially richer, more accurate foundation of raw material to explore.

- **Grounding Complex Structural Topologies:** Both the Adversarial Refinement Agent (47.00% $\rightarrow$ 55.16%) and the Hierarchical Strategic Planning Agent (45.36% $\rightarrow$ 55.32%) demonstrate robust, consistent gains of roughly 8% to 10%. For these highly structured topologies, search acts as an factual grounding.
  - *In the Adversarial setting:* External search provides concrete evidence to resolve internal debates, preventing the agents from arguing into theoretical dead ends.
  - *In the Hierarchical setting:* Search supplies the precise, step-by-step data required to successfully execute sub-goals without suffering from cascading knowledge errors.

Ultimately, this analysis demonstrates that search augmentation does more than just "add facts"—it fundamentally repairs the specific structural weaknesses inherent to different reasoning agents, elevating the baseline quality of the candidates passed to the final Synthesis agent. Overall, the full PoTRE system improves from 49.92% (Closed) to 58.40% (Open) but the theoretical upper bound (Oracle) with Search is **64.6%** (1,615/2,500), indicating significant headroom for improved synthesis strategies in future work.

5.5.1 Robustness and Boundary Conditions

We evaluate PoTRE on two distinct formats to verify the universality of these gains.

1. Closed-Ended Reasoning (MCQ Comparison). In Table 6, we specifically evaluated PoTRE on the HLE Multiple-Choice subset ($N = 591$) to enable a direct comparison with MonoScale Shao et al. (2026), a recent study that benchmarks agentic scaling on this exact partition. This allows us to isolate the impact of our reasoning topology against their strategy of simply increasing agent count.

2. Domain-Specific Expert Reasoning (PRBench Finance). To test the boundaries of search augmentation, we evaluated PoTRE on the "Hard Subset" of PRBench Finance ($N = 300$) [Table 7].

The minimal gains on PRBench provide a crucial insight: access to the internet does not automatically solve expert-level problems. While search helps significantly on open-domain knowledge tasks (HLE +8.00%), it offers little advantage on tasks requiring synthesized professional judgment. Notably, the Scaffolding Lift

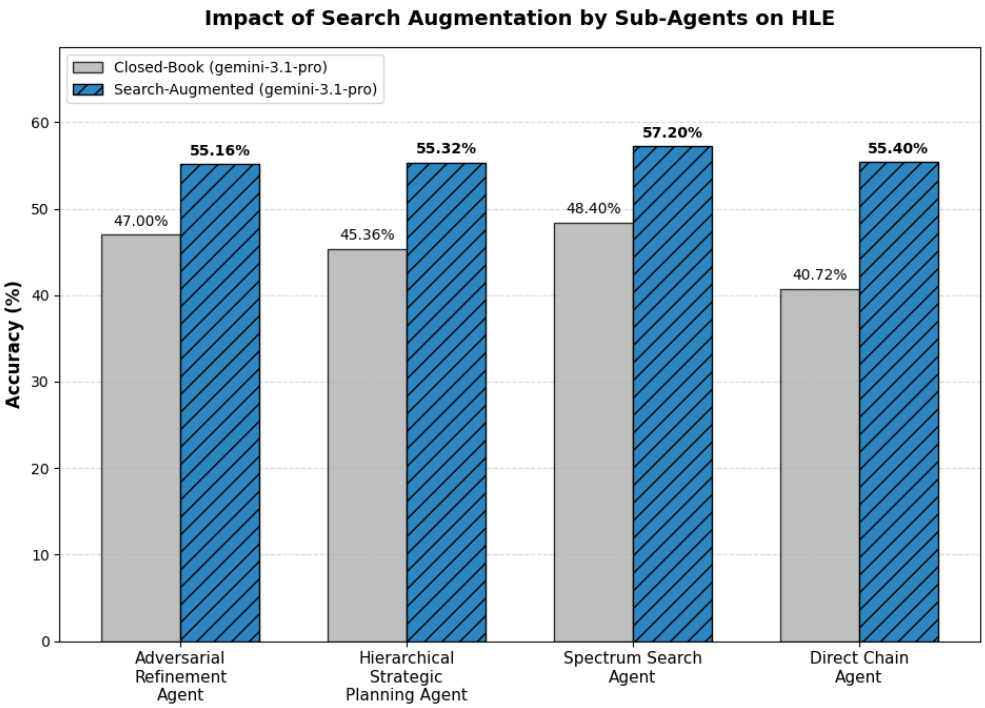

Figure 3: **Impact of Search Augmentation on HLE.** Performance comparison of the four orthogonal sub-agents under Closed-Book and Search-Augmented constraints. External retrieval universally elevates accuracy across all reasoning topologies. Notably, the Direct Chain Agent experiences the most dramatic marginal gain (+14.60%), effectively utilizing search to overcome its structural limitations and matching the performance of more complex architectural agents.

Table 7: **PRBench Control Experiment.** Search yields minimal gains (< 0.01) on this expert reasoning task. This confirms that for high-level professional domains, the bottleneck is *judgment* (reasoning), not just *information access* (retrieval).

Model	Mode	Avg Clip Score
Gemini-3-Pro-Preview	Open-Book	0.3351 (+0.0032)
Gemini-3-Flash-Preview	Open-Book	**0.3509 (+0.0023)**

persists: the lightweight Flash model (0.3509) continues to outperform the Pro model (0.3351), confirming that our agentic topology is effective even in domain-specific settings.

5.6 Orthogonality Analysis

To understand the efficacy of PoTRE's cognitive heterogeneity, we analyzed the inter-agent orthogonality and marginal utility of our four primary reasoning agents. The analysis revealed significant divergence in the failure modes of the individual agents, demonstrating that their unique topological structures actively navigate the language model toward distinct, valid regions of the solution space.

Rather than relying on mere prompt variations or homogeneous sampling, PoTRE deploys a set of structurally orthogonal reasoning agents: the Adversarial Refinement Agent performs debate, the Hierarchical Strategic Planning Agent utilizes structured diagnostic planning, and the Spectrum Search Agent synthe-

Table 8: **Cumulative Marginal Impact on HLE.** Sequential addition of each reasoning topology demonstrates the value of cognitive heterogeneity. Each distinct agent successfully solves a unique subset of problems that prior agents failed on, culminating in an upper-bound ensemble accuracy of 64.60% ($N = 2500$).

Sub-Agents	New Tasks Added	Cumulative Total	Ensemble Accuracy
Spectrum Search Agent	+1430	1430	57.20%
Direct Chain Agent	+100	1530	61.20%
Hierarchical Strategic Planning Agent	+58	1588	63.52%
Adversarial Refinement Agent	+27	1615	64.60%

Table 9: **Inter-Agent Marginal Complementarity on the HLE.** This table displays the marginal complementarity between parallel reasoning agents. Each cell indicates the percentage of questions failed by the row's agent but successfully solved by the column's complementing agent. The consistently high complementarity rates across all combinations demonstrate the strong error orthogonality of our chosen cognitive topologies.

Failed Agent \ Complementing Agent	Hierarchical Strategic Planning Agent	Adversarial Refinement Agent	Spectrum Search Agent	Direct Chain Agent
Hierarchical Strategic Planning Agent	–	11.7%	13.7%	12.8%
Adversarial Refinement Agent	12.1%	–	13.0%	12.6%
Spectrum Search Agent	9.9%	8.8%	–	9.4%
Direct Chain Agent	12.7%	12.1%	13.0%	–

sizes parallel, independent perspectives. As illustrated in Table 8, the Spectrum Search Agent establishes a strong individual accuracy of 57.20%. However, the sequential aggregation of the remaining agents yields a monotonically increasing cumulative accuracy. Notably, the *Direct Chain Agent* successfully resolves an additional 100 unique tasks that the primary spectrum search agent failed to solve. In a sub-agent level analysis evaluating the union of successful predictions—where a task is considered solved if *any* of the four sub-agents produces the correct answer—this aggregate coverage reaches 64.60%, representing a substantial absolute gain of 7.40% over the best individual constituent.

To formalize this reasoning orthogonality, we computed an Inter-Agent Marginal Complementarity matrix (Table 9), defined mathematically as the conditional probability $P(\text{Agent B solves} \mid \text{Agent A fails})$. These off-diagonal probabilities remain robust across all pairings. While the reasoning agents share a substantial baseline of knowledge—resulting in redundancy on easier tasks—their failure boundaries remain strongly orthogonal. For example, when the *Spectrum Search Agent* fails, the *Hierarchical Strategic Planning Agent* independently complementing 9.9% of those failures, while the *Direct Chain Agent* simultaneously complements 9.4%. Furthermore, our sequential ablation demonstrates that every individual agent contributed a meaningful margin on top of the growing ensemble (*Direct Chain*: +100, *Hierarchical*: +58, *Adversarial*: +27). This marginal complementarity capability empirically confirms that the reasoning manifolds of each agents are fundamentally distinct, definitively validating the necessity of a functionally diverse, "width-over-depth" architecture to prevent cognitive mode collapse.

ARC AGI 2 Benchmark Validations To prove cross-domain robustness, we identically computed the marginal impact (Table: 10) and marginal complementarity matrix (Table 11) on the ARC-AGI 2 dataset ($N = 120$ tasks). The cognitive heterogeneity of the four agents is not restricted to linguistic tasks; the architecture demonstrates equivalent efficacy when applied to the spatial grid reasoning of ARC-AGI. generating

Table 10: **Cumulative Marginal Impact on ARC-AGI-2 (Gemini-3-Flash-Preview).** Sequential addition of each reasoning topology further demonstrates the value of cognitive heterogeneity on spatial logic tasks, peaking at an ensemble accuracy of 44.2%.

Sub-Agents	New Tasks Added	Cumulative Total	Ensemble Accuracy
Spectrum Search Agent	+35	35	29.2%
Chain-of-Thought	+10	45	37.5%
Hierarchical Strategic Planning Agent	+5	50	41.7%
Adversarial Refinement Agent	+3	53	44.2%

Table 11: **Inter-Method Marginal Complementarity on ARC-AGI-2.** This table displays the marginal complementarity across all parallel reasoning agents. Each cell indicates the percentage of questions failed by the row's agent but successfully solved by the column's complementing agent. Noticeably, the Spectrum Search Agent serves as a massive rescuer for the other topologies on this spatial benchmark.

Failed Agent \ Complementing Agent	Hierarchical Strategic Planning Agent	Direct Chain Agent	Spectrum Search Agent	Adversarial Refinement Agent
Hierarchical Strategic Planning Agent	–	24.3%	30.4%	2.6%
Direct Chain Agent	5.4%	–	18.5%	3.3%
Spectrum Search Agent	5.9%	11.8%	–	3.5%
Adversarial Refinement Agent	4.3%	23.9%	29.9%	–

incredibly high Marginal Complementaritys (e.g. the *ASpectrum search* agent successfully complementing nearly a third, 29.9%, of tasks failed by *Adversarial refinement agent*).

5.7 The Verification Bottleneck

Our ablation studies reveal a consistent gap between the "Oracle" potential and final performance across domains. On HLE (Open-Book), the system generated correct candidates for 64.60% of tasks, yet recovered 58.40%. Similarly, on ARC-AGI-2, the Oracle score reached 44.16% (53/120) compared to the actual score of 38.30%. This indicates that while PoTRE excels at *generating* diverse valid reasoning paths, the Synthesizer's ability to *discriminate* truth from plausible distractors remains the primary constraint.

6 Computational Cost

A critical finding of this study is the consistent "Scaffolding Lift" observed across all evaluated benchmarks (ARC-AGI-2, HLE, and PRBench). Our results demonstrate that the smaller, more efficient Gemini-3-Flash-Preview, when equipped with the PoTRE agentic framework, consistently outperforms the larger Gemini-3-Pro-Preview operating with prior work prompts Zhang et al. (2025b) and Zhang et al. (2025a). This confirms that PoTRE's gains are structural rather than merely a result of prompting disparities.

These findings challenge the assumption that reasoning performance scales strictly with model size. Instead, they suggest that for complex tasks—ranging from the visual abstractions of ARC-AGI-2 to the semantic logic of HLE—architectural scaffolding is a more decisive performance driver than raw parameter count. By offloading cognitive load to specialized agentic roles (e.g., adversarial verification, spectrum search, hierarchical planning and direct chain), PoTRE enables smaller models to exceed their parameter-bound capabilities, effectively trading internal scale for external test-time compute.

To excel at complex reasoning tasks, PoTRE effectively trades token consumption for accuracy. By employing parallel test-time compute across four distinct agents, the framework enables standard LLMs to achieve

state-of-the-art performance previously reserved for much larger models. While this agentic approach incurs a 15× token overhead compared to baseline CoT, the impact on user experience is minimized through asynchronous execution. Because the agents run in parallel, the total latency is bounded only by the slowest single component, delivering high-fidelity reasoning without prohibitive delays.

To rigorously evaluate the computational efficiency of our framework against existing literature, we calculated the inference costs based on the official Google Gemini API pricing tiers (Standard API, $\leq$ 200k context window). Because modern reasoning models utilize hidden "thinking" budgets, output costs were calculated by subtracting the logged prompt tokens from the total tokens to capture the true billed output volume (which encompasses both visible candidate tokens and hidden reasoning tokens).

We logged exact token consumption for our evaluations across the 2,158-question text-only subset of HLE.

- **Gemini 3.1 Pro Preview Variant:** Priced[2] at $2.00 per 1M input tokens and $12.00 per 1M output tokens. The evaluation consumed 28.6M input tokens and 463.2M output tokens (comprising 18.8M candidate tokens and 444.4M reasoning tokens). This resulted in an input cost of $57.13 and an output cost of $5,558.38, for a total empirical cost of $5,615.51.
- **Gemini 3 Flash Preview Variant:** Priced at $0.50 per 1M input tokens and $3.00 per 1M output tokens. The evaluation consumed 28.3M input tokens and 680.5M output tokens (19.4M candidate tokens and 661.1M reasoning tokens). Despite utilizing a larger reasoning budget to achieve its performance, the cost-efficient architecture resulted in an input cost of $14.13 and an output cost of $2,041.55, for a total empirical cost of $2,055.68.

ReThinker Baseline: Because the exact token consumption of the ReThinker framework is unpublished, we established an empirically-aligned cost baseline grounded in their reported architectural constraints. ReThinker relies on the Gemini 3 Pro model (which shares the exact $2.00/$12.00 pricing tier as Gemini 3.1 Pro Preview) and utilizes a 3-stage agentic loop (Solver, Critic, Selector). Crucially, their methodology explicitly defines a 128K maximum context window limit for their generation process.

To ensure fair comparison, we anchored the baseline to our empirical observations for the Gemini 3.1 Pro model. We matched their input token volume exactly to our empirical run (28.6M total input tokens), as both frameworks process the identical dataset text. Similarly, we established a baseline of 18.8M visible candidate tokens. To account for the generative overhead of their multi-round iterative synthesis, we allocated a highly conservative budget of ∼197M thinking tokens, yielding a total estimated output volume of 215.8M tokens (100,000 output tokens per question). This models a scenario where their agentic loop utilizes ∼113,236 total tokens per question, keeping it comfortably within their stated 128K context window limit. Applying the standard pricing model, this utilization incurs an estimated input cost of $57.13 and an output cost of $2,589.60, yielding an estimated total evaluation cost of $2,646.73. As detailed in Table 12, our Gemini 3 Flash Preview variant surpasses the baseline's reported accuracy (55.28% vs. 52.20%). Crucially, despite generating nearly triple the volume of raw reasoning tokens to achieve this performance (680.5M vs. an estimated 215.8M), our framework's total cost remains modest at $2,055.68 vs. an estimated $2,646.73. This demonstrates that leveraging a high-throughput model like Gemini 3 Flash Preview allows for significantly deeper, more exhaustive reasoning trajectories—ultimately outperforming complex multi-agent Pro-level baselines—while maintaining strict cost superiority.

7 Conclusion

We introduce PoTRE, a heterogeneous multi-agent architecture that synthesizes adversarial, hierarchical, spectrum search and direct chain agents to solve complex general intelligence tasks. By decoupling reasoning into specialized cognitive roles—verification, planning, and search—we demonstrate that architectural topology is as critical as model scale for high-order problem solving.

PoTRE establishes a new state-of-the-art on *Humanity's Last Exam*, achieving **49.92%** in closed-book settings and **58.40%** in open-book mode on full set where it outperforms all the existing models and agentic

[2]Official Pricing: `https://ai.google.dev/gemini-api/docs/pricing`

Table 12: **Accuracy, Token Consumption, and Estimated Cost Comparison.** Evaluated on the HLE Text-Only Subset ($N = 2,158$). The Flash variant of our framework demonstrates exceptional cost-efficiency, outperforming the baseline in accuracy while costing significantly less to run.

Metric	Baseline (Empirically-Aligned)	Our Framework (Empirical)	
	ReThinker† (Gemini-3-Pro-Preview Preview)	**Flash Variant** (Gemini-3-Flash-Preview Preview)	**Pro Variant** (Gemini-3.1-Pro-Preview Preview)
Accuracy	52.20%	55.28%	**60.24%**
Input Tokens	∼28.6M	28.3M	28.6M
Output Tokens*	∼215.8M	680.5M	463.2M
Total Tokens	∼244.4M	708.8M	491.8M
Estimated Cost	∼$2,646.73	**$2,055.68**	$5,615.51

* Output encompasses both visible candidate tokens and model-generated reasoning/thinking tokens.

† ReThinker tokens are estimated by matching our empirical input volume and utilizing their stated 128K context limit.

frameworks. Furthermore, PoTRE achieves 60.24% on text only subset, outperforming frameworks like Yunque DeepResearch (51.7%) and ReThinker (52.2%). We observe a robust "Scaffolding Lift" across all three benchmarks: the lightweight Gemini-3-Flash-Preview backbone, when augmented with PoTRE, consistently outperforms the larger Gemini-3-Pro-Preview model—achieving absolute superiority on ARC-AGI-2 (38.30% vs. 21.66%) and PRBench (0.3486 vs. 0.2603)—and effectively matches or exceeds even high-cost Self-Consistency ensembles ($N = 16$). This suggests that for tool-augmented tasks, the *latency-throughput* advantages of efficient models—leveraging cognitive heterogeneity to stabilize massive parallel exploration—outweigh both the parametric depth of frontier models and the statistical benefits of homogeneous sampling.

Two critical objectives are highlighted for future investigation. First, priority is placed on closing the "The Verification Bottleneck" (64.60% Oracle vs. 58.40% Actual), where correct candidates currently generated are frequently overlooked by the Synthesis agent. To resolve this, a transition moves beyond black-box evaluation by incorporating process-level analysis, ensuring that underlying generative logic is verified rather than relying on surface-level plausibility. Second, the implementation of adaptive agent routing is identified as a necessary advancement. By utilizing a lightweight dispatcher to dynamically activate reasoning agents based on problem difficulty, computational efficiency can be optimized while preserving ensemble robustness—essential steps in the shift from simple prompting to structured, self-correcting cognitive architectures.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Mikołaj Podstawski, Hubert Niewiadomski, Piotr Nytko, et al. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Yuxuan Cai, Xinyi Lai, Peng Yuan, Weiting Liu, Huajian Li, Mingda Li, Xinghua Wang, Shengxie Zheng, Yanchao Hao, Yuyang Yin, and Zheng Wei. Yunque deepresearch technical report, 2026. URL `https://arxiv.org/abs/2601.19578`.

Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, Jinsung Yoon, and Sercan O Arik. Sets: Leveraging self-verification and self-correction for improved test-time scaling. *arXiv preprint arXiv:2501.19306*, 2025a.

Yongchao Chen, Jiefeng Chen, Rui Meng, Ji Yin, Na Li, Chuchu Fan, Chi Wang, Tomas Pfister, and Jinsung Yoon. Tumix: Multi-agent test-time scaling with tool-use mixture. *arXiv preprint arXiv:2510.01279*, 2025b.

François Chollet and the ARC Prize Team. Arc-agi-2: A new challenge for frontier ai reasoning systems. https://arcprize.org, 2025. Accessed: 2026-01-31.

Google DeepMind. Gemini 3 pro technical report. Technical report, Google, 2025. arXiv preprint.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist, 2025. URL https://arxiv.org/abs/2502.18864.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Dan Hendrycks, Mantas Mazeika, Thomas Woodside, et al. Humanity's last exam: A benchmark for frontier ai. *arXiv preprint*, 2025. URL https://cais.org/hle.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

OpenAI. Gpt-4 technical report. Technical report, OpenAI, 2024. URL https://arxiv.org/abs/2303.08774.

Scale AI Research. Prbench: Evaluating professional reasoning in finance and law. *Scale AI Technical Report*, 2025. Internal benchmark release.

Shuai Shao, Yixiang Liu, Bingwei Lu, and Weinan Zhang. Monoscale: Scaling multi-agent system with monotonic improvement, 2026. URL https://arxiv.org/abs/2601.23219.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Hongjin Su et al. Toolorchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*, 2025.

Zhentao Tang, Yuqi Cui, Shixiong Kai, Wenqian Zhao, Ke Ye, Xing Li, Anxin Tian, Zehua Pei, Hui-Ling Zhen, Shoubo Hu, Xiaoguang Li, Yunhe Wang, and Mingxuan Yuan. Rethinker: Scientific reasoning by rethinking with guided reflection and confidence control, 2026. URL https://arxiv.org/abs/2602.04496.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, Xuanzhong Chen, Yida Zhao, Zhen Zhang, Zhengwei Tao, Zhongwang Zhang, Zile Qiao, Chenxi Wang, Donglei Yu, Gang Fu, Haiyang Shen, Jiayin Yang, Jun Lin, Junkai Zhang, Kui Zeng, Li Yang, Hailong Yin, Maojia Song, Ming Yan, Minpeng Liao, Peng Xia, Qian Xiao, Rui Min, Ruixue Ding, Runnan Fang, Shaowei Chen, Shen Huang, Shihang Wang, Shihao Cai, Weizhou Shen, Xiaobin Wang, Xin Guan, Xinyu Geng, Yingcheng Shi, Yuning Wu, Zhuo Chen, Zijian Li, and Yong Jiang. Tongyi deepresearch technical report, 2025. URL https://arxiv.org/abs/2510.24701.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Beichen Zhang, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Think visually, reason textually: Vision-language synergy in arc, 2025a. URL `https://arxiv.org/abs/2511.15703`.

Hongwei Zhang, Ji Lu, Shiqing Jiang, Chenxiang Zhu, Li Xie, Chen Zhong, Haoran Chen, Yurui Zhu, Yongsheng Du, Yanqin Gao, Lingjun Huang, Baoli Wang, Fang Tan, and Peng Zou. Co-sight: Enhancing llm-based agents via conflict-aware meta-verification and trustworthy reasoning with structured facts, 2025b. URL `https://arxiv.org/abs/2510.21557`.

A Appendix A: Architectural Decisions & Prompt Engineering

This appendix details the exact system prompts and interaction protocols used in PoTRE.

A.1 Note on Spectrum Search Agent Strategy

Architectural Note: The prompts for the spectrum search agent differ significantly between domains. This is a deliberate "Task-Adaptive" design choice to address the specific constraints of verification:

- **For ARC-AGI-2 (Exact Logic):** We generate N[3] parallel independent candidates and employ an Iterative Refinement Loop (Hypothesis → Critique → Code).
  - *Reasoning:* Since the solution space is deterministic and executable, the agent must verify its logic against training examples before final implementation.
  - *Selection:* We use output majority voting on the executed grids, as consensus is a robust signal for objective correctness.
- **For HLE & PRBench (Open-Ended):** We generate N parallel independent candidates.
  - *Reasoning:* Because these tasks lack verifiable input-output examples at inference time (unlike ARC), we cannot programmatically test and refine the answers. Instead, we rely entirely on the breadth and diversity of the generated reasoning paths.
  - *Selection:* We use an LLM judge.

The following sections provide the prompts for these components.

A.2 Humanity's Last Exam

A.2.1 Adversarial Refinement Agent via Debate (HLE)

This section contains the exact system prompts and interaction templates used in the adversarial debate loop.

System Prompt: Proposer Agent

```
PROPOSER_SYSTEM = """You are an expert Proposer in a refined debate.
Your goal is to answer the user's question as accurately as possible.
You will receive a Question (and optional Image).
You must provide a tentative Answer and a detailed Rationale.
Later, you might receive critique from a Verifier.
```

[3]For all datasets in our implementation, we set $N = 8$.

```
You must update your answer if the critique is valid.

Output JSON:
{
"rationale": "...",
"answer": "..."
}
"""
```

System Prompt: Verifier Agent

```
VERIFIER_SYSTEM = """You are an expert Verifier. You will receive a Question,
an Image (optional), and a Candidate Answer from a Proposer.
Your goal is to find flaws, logical gaps, or factual errors in the
Candidate Answer.
If the answer is correct, you should APPROVED it.
If incorrect, provide specific, constructive critique to help the Proposer fix it.

Output JSON:
{
"critique": "...",
"status": "APPROVED" or "REJECTED"
}
"""
```

Round 1: Initial Proposal, The Proposer is fed the question (and optionally the image).

Input Template: Initial Proposal

```
Question: {Question Text}
[Image] (Optional)
```

Round N: Verification, The Verifier receives the original context plus the Proposer's latest candidate.

Input Template: Verification

```
[Image] (Optional)
Question: {Question Text}

Candidate Answer: {Answer}
Candidate Rationale: {Rationale}

Critique this.
```

Round N: Refinement (Rebuttal), If the Verifier rejects the answer, the Proposer receives the critique to guide refinement.

Input Template: Refinement

```
The Verifier provided this critique:
{Critique}
```

```
Please refine your answer and rationale.
```

A.2.2 The Hierarchical Strategic Planning Agent (HLE)

This section contains the exact system prompts and interaction templates used in the Hierarchical Strategic planning agent.

System Prompt: Architect Agent

```
ARCHITECT_SYSTEM = """You are a Lead Architect for complex reasoning tasks.
Your goal is to break down the user's question into 2-3 logical sub-steps or key aspects that
     need to be analyzed to reach the correct answer.
Do NOT try to answer the question directly effectively. Instead, Plan.

Output JSON:
{
  "plan_steps": ["Step 1...", "Step 2..."],
  "reasoning_strategy": "..."
}
"""
```

System Prompt: Engineer Agent

```
ENGINEER_SYSTEM = """You are a Senior Engineer.
You will receive a plan from the Architect and the original Question (and optional Image).
Execute the plan. detailed.

Output JSON:
{
  "execution_trace": "...",
  "draft_answer": "..."
}
"""
```

System Prompt: Supervisor Agent

```
SUPERVISOR_SYSTEM = """You are a Chief Supervisor.
You have the Architect's Plan and the Engineer's Execution/Draft.
Your job is to specific the final answer and rationale, ensuring strict adherence to the
    facts and logic.
Refine the Engineer's draft if needed.

Output JSON:
{
  "rationale": "...",
  "answer": "..."
}
"""
```

System Prompt: Meta-Reviewer (Diagnosis)

```
You are a Senior Meta-Reviewer for an AI Reasoning System.
The Agent is stuck in a loop trying to solve a complex reasoning task.

HISTORY OF ATTEMPTS:
{History Summary}

YOUR DIAGNOSIS TASKS:
1. **Pattern Recognition:** Is the agent repeating the same failed logic or plan?
2. **Error Analysis:** - Is the Architect proposing plans that are too vague?
   - Is the Engineer failing to execute specific steps?
   - Is the Supervisor missing critical constraints in the final answer?
3. **Strategic Pivot:** Dynamically synthesize a high-level strategic directive to re-orient
    the Architect's approach.


OUTPUT FORMAT (JSON):
{
    "diagnosis": "The agent keeps trying to calculate X without considering Y...",
    "custom_advice": "You are ignoring the constraint about 'without moving queens'. Focus on
         knight moves."
}
```

Interaction Templates

The following templates define how data flows between the agents.

1. Architect Step Input

```
[Image] (Optional)
Question: {Question Text}

(Note: If Meta-Reviewer intervenes, a "CRITICAL INTERVENTION" block is appended here)
```

2. Engineer Step Input

```
[Image] (Optional)
Question: {Question Text}

Architect's Plan:
{Plan JSON}

Execute this plan detailed.
```

3. Supervisor Step Input

```
[Image] (Optional)
Question: {Question Text}

Architect Plan: {Plan JSON}
Engineer Draft: {Draft Answer JSON}
```

```
Provide the Final Answer.
```

A.2.3 Spectrum Search Agent (HLE)

This section contains the exact system prompts and interaction templates used in the Spectrum Search Agent.

System Prompt: Candidate Generator

```
CANDIDATE_SYSTEM = """You are a Candidate Generator.
You will receive a Question (and optional Image).
Propose a detailed, logical answer. Think step by step.
Output JSON: { "rationale": "...", "answer": "..." }
"""
```

System Prompt: Expert Judge

```
JUDGE_SYSTEM = """You are an Expert Judge.
You have a Question and multiple Candidate Answers.
Your job is to select the BEST answer based on accuracy, logical soundness, and completeness.
Output JSON: { "best_candidate_index": int, "reason": "..." }
"""
```

Interaction Templates

Candidate Generation Step Each worker receives the same input independently.

Input: Candidate Generation

```
[Image] (Optional)
Question: {Question Text}
```

Judging Step The Judge receives the original question and the aggregated list of candidates.

Input: Judging Step

```
Question: {Question Text}

--- Candidate 0 ---
Rationale: {Rationale Text}
Answer: {Answer Text}

--- Candidate 1 ---
Rationale: {Rationale Text}
Answer: {Answer Text}

...

Select the index (0-based) of the best candidate.
```

A.2.4 Direct Chain Agent (HLE)

System Prompt: CoT Instruction

```
SYSTEM_INSTRUCTION = """You are an expert in humanities, law, and logical reasoning.
Answer the following question.
You must output your answer in valid JSON format:
{
  "rationale": "Detailed step-by-step reasoning...",
  "answer": "The final answer (e.g. 'A', 'B', '42', 'summary')"
}
"""
```

Interaction Template The agent performs a single-turn generation.

Input Template

```
[Image] (Optional)
Question: {Question Text}

Provide your detailed rationale and final answer in JSON.
```

A.3 Financial Domain Prompts

A.3.1 Adversarial Refinement Agent (Finance)

This section contains the exact system prompts and interaction templates used in the financial adversarial debate loop.

- **Proposer Agent**

System Prompt: Financial Architect

```
You are a Senior Financial Architect.
Your goal is to answer the user's finance question as accurately as possible.
You will receive a conversation history or a question.

Provide a comprehensive, accurate, and professional answer.
```

- **Verifier Agent**

System Prompt: Financial Auditor

```
You are a Senior Financial Auditor.
You will receive a Question and a Candidate Answer from a Proposer.

Your goal is to verify the answer for:
1. Accuracy: Are the financial facts and calculations correct?
2. Completeness: Did the answer address the user's question fully?

If the answer is accurate and complete, APPROVED it.
If not, provide specific critique demanding the missing or incorrect elements.
```

```
Output JSON:
{
  "critique": "...",
  "status": "APPROVED" or "REJECTED"
}
```

- **Dynamic Context & Flow** The context is constructed linearly. The agent sees the following sequence:

  1. System Instruction (Defined above for Proposer/Verifier)

  2. Conversation History (Injected First) All previous turns are passed to the model's history before the current question.

  Context Structure

```
[User]: {prompt_0}
[Assistant]: {response_0}
...
[User]: {prompt_N}
[Assistant]: {response_N}
```

  3. Current Turn After the history, the current tasks are appended.

  Input: Proposer View

```
{current_question}
[Optional Image Attachment]
```

  The verifier sees the history, then receives the Current Question bundled with the Candidate Answer in a single turn.

  Input: Verifier View

```
{current_question}
[Optional Image Attachment]

Candidate Answer: {current_answer}
Candidate Rationale: {current_rationale}

Critique this.
```

  If the Verifier rejects the answer, the Proposer receives the critique to guide the refinement.

  Input: Refinement Instruction

```
The Verifier provided this critique:
{critique}

Please refine your answer and rationale.
```

A.3.2 Hierarchical Strategic Planning Agent (Finance)

This section contains the prompts for the hierarchical financial reasoning agents.

- **Architect Agent**
  - **Role:** Principal Financial Architect
  - **Goal:** Break down the problem into a logical plan.

System Prompt: Architect

```
You are a Principal Financial Architect.
Your goal is to break down the user's finance problem into 2-3 logical sub-steps (e.g
    . Regulatory Analysis, Quantity Calculation, Risk Assessment).
Do NOT answer the question directly. Instead, Plan.

Output JSON:
{
  "plan_steps": ["Step 1...", "Step 2..."],
  "reasoning_strategy": "..."
}
```

Context: Receives the full conversation history (User/Assistant turns).

- **Engineer Agent**
  - **Role:** Senior Financial Analyst
  - **Goal:** Execute the Architect's plan rigorously.

System Prompt: Engineer

```
You are a Senior Financial Analyst.
You will receive a plan from the Architect and the original Question.
Execute the plan rigorously. Use financial formulas and regulations where applicable.

Output JSON:
{
  "execution_trace": "Detailed work...",
  "draft_answer": "..."
}
```

Appended Instruction: Engineer Input

```
Architect's Plan: {plan_json}

Execute this plan to answer the previous question.
```

Context: Receives the full conversation history.

- **Supervisor Agent**
  - **Role:** Chief Risk Officer (CRO)
  - **Goal:** Synthesize the final answer from the plan and execution draft.

System Prompt: Supervisor

```
You are a Chief Risk Officer (CRO).
You have the Architect's Plan and the Analyst's Execution/Draft.
Your job is to synthesize the Final Answer.

Ensure the answer is comprehensive, accurate, and professional.
Refine the draft to be technically precise and practically actionable.

Output JSON:
{
  "answer": "..."
}
```

Appended Instruction: Supervisor Input

```
Architect Plan: {plan_json}
Engineer Draft: {eng_json}

Review the draft and provide the Final Answer.
```

Context: Receives the full conversation history.

System Prompt: Meta-Reviewer (Diagnosis)

```
You are a Senior Meta-Reviewer for an AI Reasoning System.
The Agent is stuck in a loop trying to solve a complex reasoning task.

HISTORY OF ATTEMPTS:
{History Summary}

YOUR DIAGNOSIS TASKS:
1. **Pattern Recognition:** Is the agent repeating the same failed logic or plan?
2. **Error Analysis:** - Is the Architect proposing plans that are too vague?
   - Is the Engineer failing to execute specific steps?
   - Is the Supervisor missing critical constraints in the final answer?
3. **Strategic Pivot:** Dynamically synthesize a high-level strategic directive to re-orient
    the Architect's approach.

OUTPUT FORMAT (JSON):
{
    "diagnosis": "The agent keeps trying to calculate X without considering Y...",
    "custom_advice": "You are ignoring the constraint about 'without moving queens'. Focus on
         knight moves."
}
```

A.3.3 Spectrum Search Agent (Finance)

This section details the parallel generation and judging prompts used for financial queries.

- Candidate Agent
  - Role: Quantitative Financial Analyst

  - Goal: Generate a top-tier answer. (Count: 8 parallel candidates).

System Prompt: Candidate

```
You are a Quantitative Financial Analyst candidate.
You will receive a Finance Question or Context.

Your goal is to provide a "Top Tier" answer that is accurate, specific, and professional.
Focus on explaining the concepts clearly and citing relevant examples.
```

Context: Receives the full conversation history. Ends with the current User Question (and optional image).

- Judge Agent
  - Role: Chief Investment Officer (CIO)
  - Goal: Select the best answer from the candidates.

System Prompt: Judge (CIO)

```
You are a Chief Investment Officer (CIO) and Portfolio Manager.
You have a Question and multiple Candidate Answers from your analysts.

Your job is to select the BEST answer based on:
1. **Mechanistic Depth**: Does it explain the root cause/driver?
2. **Specificity**: Does it use real-world examples/regulations?
3. **Risk Awareness**: Does it cover downside risks?
4. **Clarity**: Is it easy to understand?

Output JSON: { "best_candidate_index": int }
```

Input Template: Judge's View The Judge receives the full conversation history (User turns + final question). Then, the candidate answers are appended to the final user turn.

Input: Judge Context

```
[History...]

[User]: {current_question}
[Optional Image Attachment]

Here are the candidate answers from your analysts:

--- Candidate 0 ---
Answer: {answer_0}

--- Candidate 1 ---
Answer: {answer_1}

...

--- Candidate n ---
Answer: {answer_n}

Select the index (0-based) of the best candidate.
```

A.3.4 Direct Chain Agent (Finance)

- Role: Distinguished Professor of Finance
- Goal: Derive answers from first principles using logical steps.

System Prompt: Direct Chain Agent

```
You are a Distinguished Professor of Finance.
The user will provide a Finance Question or Context.
You must derive the answer from first principles.

Break down complex problems into simple, logical steps. Use analogies where helpful.
```

Dynamic Context & Flow The context is constructed linearly. The agent sees the following sequence:

A. Conversation History (Injected First) All previous turns are passed to the model's history *before* the current question.

Context: Conversation History

```
[User]: {prompt_0}
[Assistant]: {response_0}
...
[User]: {prompt_N}
[Assistant]: {response_N}
```

B. Current Turn After the history, the current tasks are appended.

Context: Current Turn

```
{current_question}
[Optional Image Attachment]
```

A.4 Neuro-Symbolic (ARC-AGI-2)

A.4.1 Adversarial Refinement Agent (ARC-AGI)

- Vision Agent (Proposer) Prompts
  - Initial Observation (First Training Pair)
  - Context: The agent sees the first input/output pair.

Input: Initial Observation

```
Here is the first training pair for an ARC task.
Input Grid:
[NUMPY_ARRAY]
Output Grid:
[NUMPY_ARRAY]
[ATTACHED IMAGE: Input/Output rendered]
```

```
What do you see? Describe the objects, potential transformation, and pattern.
```

- Subsequent Observations
  - Context: The agent sees pairs $2 \ldots N$.

Input: Subsequent Observation

```
Here is pair [INDEX].
Input Grid:
[NUMPY_ARRAY]
Output Grid:
[NUMPY_ARRAY]
[ATTACHED IMAGE: Input/Output rendered]

What do you see now? Does this confirm or change your previous hypothesis?
```

- Rule Consolidation (Pre-Debate)
  - Context: After observing all pairs, the agent must commit to a rule.

Instruction: Rule Consolidation

```
Based on all the examples you have seen, strictly state your
FINAL TRANSFORMATION RULE.
```

- Verifier Agent (Critic) Prompts
  - Initial Verification
  - Context: Agent 2 receives all training data and Agent 1's proposed rule.

System Prompt: Initial Verification

```
I am sharing all the training inputs for this ARC task with you.
--- Pair 1 ---
Input: [GRID]
Output: [GRID]
[ATTACHED IMAGE]
... (Repeated for all pairs) ...

Another pro agent analyzed these pairs individually and proposed
this TRANSFORMATION RULE:

[AGENT_1_PROPOSED_RULE]

What do you think? Is it correct? If yes, say 'YES' or 'PASS'. I
f no, say 'NO' or 'FAIL' and provide the reason.
```

- Debate Step (Counter-Argument)
  - Context: If Agent 1 provides a rebuttal, Agent 2 is asked to re-evaluate.

Input: Re-evaluation

```
Agent 1 replied:
[REBUTTAL_TEXT]

What do you think now? PASS or FAIL?
```

- Final Code Verification
  - Context: After Agent 1 generates python code that passes training data, Agent 2 gives a final confidence vote.

Input: Final Code Check

```
Here is my final code, it passed on all train pairs.

[GENERATED_CODE]

Are you also confident that it will pass hidden test case too?
Please reply YES or NO and give your explanation.
```

- Feedback & Refinement Loops
  - Rebuttal Request (Agent 1)
  - Context: When Agent 2 rejects the rule.

Instruction: Rebuttal

```
The Verifier analyzed your rule and said:
[VERIFIER_CRITIQUE]

Please revise your thought or explain why you are correct.
```

- Code Generation Request
  - Context: Converting the agreed-upon rule to code.

Instruction: Code Generation

```
[CONTEXT_MSG or "Your transformation rule is correct..."]

Provide python code.
If you want to run an adhoc test/diagnosis, start with:
# TYPE: DIAGNOSIS

If you are confident and providing the solution, start with:
# TYPE: SOLUTION

IMPORTANT: Your ‘solve(grid)‘ function MUST return a ‘numpy.ndarray‘, NOT a list.

Return ONLY the code block.
```

- Visual Feedback (Execution Failure)
  - Context: If the code runs but produces incorrect output on training data.

Feedback: Execution Failure

```
Execution Results:
Pair [N]: FAIL - [REASON]
...

See the attached visualization of your failure (Input | Prediction | Expected).
[ATTACHED IMAGE: Side-by-side comparison of failure]
```

A.4.2 Hierarchical Strategic Planning Agent (ARC-AGI)

This section details the meta-cognitive layer used when the primary solver fails.

1. The Diagnosis Prompt This is the core prompt sent to the Meta-Reviewer when the primary solver gets stuck. It aggregates the history of failures and asks for a strategic intervention.

System Prompt: Meta-Reviewer Diagnosis

```
You are a Senior Meta-Reviewer for an AI Autonomous Coding Agent.
The Agent is stuck in a loop trying to solve an ARC task.

HISTORY OF ATTEMPTS:
[HISTORY_LOG_SUMMARY]
(e.g., "Turn 1: Failed - IndexError", "Turn 2: Failed - Robustness Error")

YOUR DIAGNOSIS TASKS:
1. **Pattern Recognition:** Is the agent repeating the same failed logic?
2. **Error Analysis (CRITICAL):** - Look for "ROBUSTNESS FAILURE" or "Color Invariance" in
    the logs.
   - If present, the Agent is hardcoding colors (e.g. `grid[mask] = 3`). You MUST stop this.

OUTPUT FORMAT (JSON):
{
    "diagnosis": "The agent is failing Color Invariance checks by hardcoding '3'.",
    "custom_advice": "Stop using magic numbers. Infer the target color from the example pairs
        ."
}
```

2. Role-Based System Prompts The Hierarchical stratrgic planning agent orchestrates three specialized sub-agents.

2.1. Lead Architect Responsible for high-level reasoning and hypothesis generation.

System Prompt: Lead Architect

```
You are the Lead Architect. Your goal is to find the Single Abstract Rule for an ARC task.

CRITICAL SUCCESS CRITERIA:
1. UNIVERSALITY: The rule must apply to ALL training pairs exactly the same way.
  - REJECT: "If Pair 1 do X, If Pair 2 do Y."
```

```
  - ACCEPT: "For ALL inputs, find the object with the most unique colors and rotate it 90 deg
      ."

2. VISUAL SCAN PRIORITY: LEGENDS & KEYS
  - Sometimes the rule is explicitly written in the grid as a "Legend" or "Key".
  - SCAN THE CORNERS: Look specifically at the corners ... for small, isolated clusters of
      pixels...
  - IF FOUND: Flag this immediately. This is likely a "Recoloring Key" or "Symbol Map."

3. REFINE LOOP: If your hypothesis requires "special cases" for different inputs, it is WRONG
    . You must discard it and REFINE the hypothesis to find the deeper, shared pattern.

Instructions:
1. Analyze Visuals: Look for 'Compass', 'Alignment', or 'Counting' patterns in the images
    first.
2. Write Universal Code: Provide a `solve(grid)` function. Always start with `grid = np.array
    (grid)`.
3. Dynamic Derivation: No magic numbers. Use `np.unique` or border scanning to find values.

Output Format (JSON): {
  "visual_observation": "(REQUIRED) Describe the object/grid alignment patterns.",
  "thought_process": "Reasoning...",
  "hypothesis": "The rule is...",
  "code": "Full python code..."
}
```

2.2. Technical Engineer Responsible for cleaning code and fixing syntax before execution.

System Prompt: Technical Engineer

```
You are the Technical Validator (Engineer). Your role is to take the python code provided by
    the Architect and prepare it for execution.

Your Responsibilities:
1. Code Assembly: Ensure the code is a complete, self-contained python block. Include `import
     numpy as np`.
2. Technical Correction: Fix syntax errors (like missing colons) and ensure `grid = np.array(
    grid)` is the first line.
3. Validation: If code fails, provide a specific diagnostic report.

Output Format: Return ONLY the cleaned python code block inside triple backticks.
```

2.3. Supervisor (Loop Prevention) Ensures the Architect doesn't propose the same failed hypothesis twice.

System Prompt: Supervisor

```
You are the SUPERVISOR for an ARC-AGI Solver System.
Your goal is to prevent the "Architect" agent from getting stuck in a feedback loop.

### INPUT DATA
1. HISTORY: A list of previously attempted hypotheses and their outcomes (Pass/Fail).
2. CURRENT_PROPOSAL: The new hypothesis the Architect just generated.
```

```
### YOUR TASK
Compare the CURRENT_PROPOSAL against the HISTORY.
Determine if the Architect is proposing a "Functionally Identical" strategy to one that
    already failed.

### DEFINITION OF "FUNCTIONALLY IDENTICAL"
Two strategies are identical if they result in the same pixel manipulation, even if the
    wording is different.

* **Example of a LOOP (REJECT THIS):**
   * History: "Crop the object and rotate 90 degrees CCW." (Failed: Pixel Mismatch)
   * Current: "Identify the red shape, extract the subgrid, and turn it Left."
   * *Verdict:* REJECT. "Turn Left" is "Rotate 90 CCW". This is the same logic.

### OUTPUT FORMAT
Return a JSON object:
{
  "status": "APPROVED" | "REJECTED",
  "reasoning": "Brief explanation of why this is new or a repeat.",
  "feedback_to_architect": "If REJECTED, write a strict instruction forcing a pivot..."
}
```

3. Interaction & Runtime Inputs These are the inputs sent to agents during specific phases of the execution loop.

3.1. Architect Input Sent at the start of Turn 1.

Input: Architect Task Data

```
TASK DATA (Images and Grids):

--- PAIR 1 ---
INPUT GRID (Visual): [IMAGE_BLOB]
INPUT GRID (Numeric):
[NUMPY_ARRAY]

OUTPUT GRID (Visual): [IMAGE_BLOB]
OUTPUT GRID (Numeric):
[NUMPY_ARRAY]

... (Repeated for all training pairs) ...
```

3.2. Supervisor Input (Loop Check) Sent every time the Architect proposes a new hypothesis.

Input: Supervisor Check

```
HISTORY:
- Attempt 1: [HYPOTHESIS] (Result: [ERROR_TYPE])
- Attempt 2: [HYPOTHESIS] (Result: [ERROR_TYPE])

CURRENT_PROPOSAL:
[NEW_HYPOTHESIS_TEXT]
```

3.3. Architect Input (Supervisor Rejection) Feedback sent if the Supervisor detects a loop.

Feedback: Supervisor Rejection

```
SYSTEM ALERT: Your plan was REJECTED by the Supervisor.
REASON: [REASONING]
INSTRUCTION: [FEEDBACK_TO_ARCHITECT]
ACTION: Propose a DIFFERENT strategy.
```

3.4. Engineer Input (Code Assembly) Sent to the Engineer to finalize the code.

Input: Engineer Code Assembly

```
ARCHITECT CODE:
[CODE_FROM_ARCHITECT]

TASK: Assemble into a robust `solve(grid)` function.
REMINDER: Start with `grid = np.array(grid)`.

CRITICAL: If the Critic suggests a specific algorithmic approach (e.g., 'Use a mask', 'Use
    flood fill'), YOU MUST ATTEMPT IT. The Critic's technical insight overrides the
    Architect's initial hypothesis if the original code is failing.
```

3.5 Code Critique and Refinement

Critique

```
You are a Logic Rigor Critic for the ARC Challenge.

    Review the Python code below.
    Your Goal: Ensure the code implements a **Single Abstract Rule** that works for ALL
        inputs universally.
```

Refinement

```
The previous code was REJECTED by the Logic Critic.

CRITIC FEEDBACK:
"{msg}"

OFFENDING CODE:
```python
{current_code}
```

TASK:
Rewrite the `solve(grid)` function to fix the violation above.
1. Address the CRITIC FEEDBACK directly.
2. If the Critic says the logic is flawed (e.g., 'Use different logic'), YOU MAY
    CHANGE THE APPROACH.
3. If the feedback is about style (e.g., 'Remove patching'), fix the
    implementation style.
4. Return the full corrected Python code.
```

Debug

```
DEBUG REPORT (Attempt {debug_count}/{max_retries}):
The code executed but produced the WRONG output.

VISUAL EVIDENCE:
Review the attached image (Left=Input, Middle=Your Prediction, Right=Expected).

DATA MISMATCH:
{log_msg}

HINT:
{hint}

PREVIOUS FAILED CODE (Do NOT repeat this logic):
```python
{current_code}
```

TASK: Write the CORRECTED Python code.
```

Hint comes from checks like comparing shapes from predicted and expected train pairs, for example, if rotated version matches expected then corresponding hint is injected.

Debug code crash

```
CRITICAL EXCEPTION (Attempt {debug_count}/{max_retries}):
The code crashed or failed robustness checks.

ERROR TRACEBACK:
{log_msg}

PREVIOUS FAILED CODE:
```python
{current_code}
```

TASK: Rewrite the code to handle this edge case or error.
Ensure color invariance and boundary checks.
```

A.4.3 Spectrum Search Agent

This section outlines the prompt sequence for the perception-induction-coding loop.

- Phase 1: Hypothesis Generation (Perception & Induction)
  - Goal: Induce the abstract transformation rule from multimodal data (Text Grids + Images + Vision JSON (computed using skimage module)).

System Prompt: Hypothesis Generation

```
You are an expert ARC-AGI Solver and python Programmer.
GOAL: Induce the transformation rule from the training examples and write a python function `
    transform(input_grid)`.
```

```
--- DATA SOURCES ---
1. RAW GRID: Use for syntax (indexing, shapes).
2. VISION JSON: Use for logic (objects, colors, counting). TRUST THIS DATA.
3. IMAGE: Use for gestalt patterns (symmetry, containment).

--- VISUAL ANALYSIS REPORT (Pre-computed JSON) ---
TRUST THIS JSON OVER YOUR OWN EYES.
[VISUAL_DESCRIPTION]

--- TRAINING EXAMPLES ---
[MULTIMODAL_DATA: Text + Image for each pair]

[ERROR_CONTEXT_IF_RETRY]

--- TEST TASK ---
INPUT GRID (RAW): ...
INPUT VISION JSON: ...
INPUT IMAGE: [IMAGE_BLOB]

--- ANALYSIS INSTRUCTIONS ---
1. ANALYZE CHANGE: Compare Input JSON vs Output JSON.
2. ASK 'WHY?':
   - Why did this object move here? (Gravity? Attraction? Alignment?)
   - Why did this color change? (Intersection? Size? Enclosure?)
3. BI-DIRECTIONAL CHECK (Mental):
   - Can you reconstruct the Input from the Output? If not, is information lost? (e.g.
       projection)
   - Describe the transformation as a precise algorithm.
   - DO NOT just predict pixels. Predict the FUNCTION.

--- RESPONSE GUIDELINES ---
1. DO NOT WRITE python CODE YET.
2. FOLLOW THE FORMAT BELOW EXACTLY.
3. Your Hypothesis must be natural language.

Format:
[Analysis]
1. Observation: ...
2. Why: ...
3. Reversibility: ...
[Hypothesis]
...
```

- Phase 2: The Logic Critic (Self-Correction & Refinement)
  - Goal: Critically evaluate the initial hypothesis and produce a "Refined Algorithm". This strictly separates reasoning from coding.
- Iteration 1 Prompt

Prompt: Critique & Refinement

```
YOUR TASK:
1. CRITIQUE: What is WRONG with your Turn 1 Hypothesis? (Edge cases? Ambiguity?)
```

```
2. SIMPLICITY CHECK: ARC tasks are usually solved by 1-3 simple core rules. If your logic is
    a page long, IT IS LIKELY WRONG. Can you simplify?
3. REFINE: precise natural language algorithm.
4. MENTAL VERIFICATION: Walk your hypothesis through ALL training examples.
   - If it fails even one pixel on Example 1, IT IS WRONG. Discard it.
4. STATUS CHECK: Are you confident this algorithm is perfect?

Format:
[Critique]
...
[Refined Algorithm]
...
[Status]
CONTINUING (or READY)
```

- Subsequent Iteration Prompt

Prompt: Critique & Refinement (Turn 2, Iteration N)

```
PREVIOUS REFINEMENT:
[CONTEXT]

YOUR TASK:
1. CRITIQUE the Previous Refinement. Did it fix the issues? Any new edge cases?
2. SIMPLICITY CHECK: Is it still too complex? Try to cut 50% of the steps. Think in terms of
    Objects and Physics (Gravity, Collision, etc).
3. REFINE again.
3. STATUS CHECK: Are you confident now?
```

- Phase 3: The Solver (Implementation)
  - Goal: Translate the validated "Refined Algorithm" into a self-contained python function.

System Prompt: The Solver

```
--- INSTRUCTIONS ---
1. Implement the solution using your Validated Hypothesis and Algorithm.
2. Simplify your logic. Focus on: Extract Object -> Transform (Rotate/Flip/Align) -> Result.
3. Write the python function ‘transform(grid)‘.
4. The code MUST be self-contained. DO NOT assume any external helper functions serve you.
5. Output ONLY valid python code in a code block.
6. Use the ‘transform‘ function signature.
```

Repair Code

```
You are an expert in solving Abstract Reasoning Corpus tasks.,
GOAL: Fix the Python Code which fails on the Training Examples.,
"--- VISUAL CONTEXT ---",
visual_description
--- TRAINING EXAMPLES (VISUAL) ---,
*image_parts,
```

```
--- CURRENT CODE (BUGGY) ---,
‘‘‘python\n{code}\n‘‘‘,
--- ERROR FEEDBACK ---,
error_feedback,
--- INSTRUCTIONS ---,
1. LOOK at the images and the error feedback.,
2. THINK STEP-BY-STEP: Analyze the Error. Is it a Logic Error, Shape Mismatch, or
    Pixel Mismatch? Why did it happen?",
3. GENERALIZATION STRATEGY: How can you fix this logic so it works for ALL examples,
    not just this specific failure?",
4. IMPLEMENTATION: Return the fixed Python code.,
5. Output Format:\n[Analysis]\n...\n[Code]\n‘‘‘python\n...\n‘‘‘
```

A.4.4 Direct Chain Agent (ARC-AGI)

- Turn 1: Analysis & Verifiable Hypothesis
  - Goal: Force the model to "show its work" by interacting with the python environment to verify hypotheses *before* committing to a solution. The model must implement a testing loop where it discards wrong hypotheses.

System Prompt: Analysis & Verification

```
I have uploaded a JSON file containing train pairs (Input/Output). I need you
    to reverse-engineer the exact transformation logic.
Do NOT guess the logic yet. Follow this strict process using python:

Analyze the Object Differences: Write a script to compare the Input grid vs
    the Output grid for the first 3 examples.
Which pixels changed?
Which pixels stayed the same?
Is there a 'separator' line (a unique column)?

Generate Hypotheses: Based on step 1, list 3 possible mathematical rules (e.g
    ., 'Mirroring', 'Periodicity based on count', 'Color mapping').

Test & Verify (Crucial):
Write a function transform(input_grid) for each of your 3 hypotheses.
Run these functions on the train inputs.
Compare your generated output to the actual train output.
If the match is not 100%, the hypothesis is WRONG. Discard it and try another.

Final Answer: Only output the logic that achieves a 100% match on all training
     examples.

--- TRAIN PAIRS ---
[JSON_DATA]
```

- Turn 2: Test Case Execution
  - Goal: If the model found a verifiable rule in Turn 1, this prompt asks it to apply that rule to the hidden test cases. It explicitly allows for a final code fix if the previous step wasn't perfect.

System Prompt: Test Execution

```
Here are the test cases. I need you to run your code on ALL of them.
Your code must handle BOTH the Training examples (perfectly) and these Test
    examples.
If your previous code was correct, reuse it. If not, fix it so it works on all
     cases.
Return the Final Code that solves these cases.

--- Test Inputs ---
[JSON_DATA]
```

A.5 Synthesis Agent Prompts

System Prompt: Humanity's Last Exam

```
"""Problem:
{problem_context}

Candidates:
{candidates_text}

Trust your instinct and pick one answer. Provide ONLY the answer text, not the candidate
    number or reference.

Final Answer:
"""
```

System Prompt: PRBench Finance

```
You are an Expert Finance Judge and Senior Auditor.
You are given multiple candidate answers to a complex finance task.
Your goal is to SYNTHESIZE a single "Super-Answer" that combines the best parts of all
    candidates.

Problem:
{problem_context}

Candidates:
{candidates_text}

Synthesis Instructions:
1. **Enrichment**: Dynamically select the best elements from ANY candidate. For example, if
    one candidate has better calculations, use them. If another has better citations or
    formatting, use those. Do not assume specific candidates have specific strengths.
2. **Completeness**: Ensure every single user question is answered with maximum depth.
3. **Accuracy**: If candidates have conflicting numbers, perform a sanity check and use the
    most plausible/consistent one.
4. **Tone**: Professional, authoritative, and precise.
5. **Output**: Provide ONLY the final merged answer. Do not explain your process.

Final Answer:
```

System Prompt: ARC-AGI-2

```
You are a Principal Logic Expert and ARC-AGI Judge.
You are tasked with determining the Single Correct Output Grid for a given Test Input.
You have access to:
1. **Problem Description**: Training examples demonstrating the rule.
2. **Candidates**: Proposed solutions from other expert agents.

GOAL: Synthesize the perfect solution.
- The candidates might be correct, partially correct, or completely wrong.
- Use the Training Examples to VERIFY the rule.
- If a candidate follows the rule perfectly, select it.
- If candidates diverge, use your reasoning to correct the errors.
- If all candidates are wrong, derive the solution from scratch.

Problem:
{problem_context}

Candidates:
{candidates_text}

OUTPUT FORMAT:
Provide ONLY the final output grid as a raw JSON list of lists (e.g. [[1,0],[0,1]]).
Do not include markdown blocks (```json) or explanations. Just the raw JSON string.

Final Answer:
```

A.6 Baseline Prompts

Humanity's Last Exam

```
   # Role and Objective
You are an expert reasoning assistant helping to solve complex problems. Your goal is to
    provide the correct answer by strictly following a rigorous thinking process.
# General Rules
1. You MUST plan extensively before generating the final answer.
2. Reflect on your reasoning at each step to catch potential errors.
3. Do not just guess; derive the answer through logical deduction.
# Task Execution Rules:
1. Analysis:
  - Break down the problem into key components.
  - Identify what information is given and what needs to be inferred.
2. Step-by-Step Reasoning:
  - Execute your plan step by step.
  - For each step, explicitly state your reasoning and the conclusion of that step.
  - If a step involves calculation or logic, double-check it.
3. Reflection:
  - Before concluding, ask yourself: "Does this make sense?", "Is there a counter-example?",
      "Did I miss any constraints?"
4. Final Output:
  - Provide the final answer clearly at the end.
  - If the question asks for a specific format (e.g., a number, a code snippet), strictly
      follow it.
# Environment Information
```

```
- You are running in a restricted reasoning environment without external tools (no web search
    , no code execution).
- Rely solely on your internal knowledge and logical capabilities.

IMPORTANT: You must output your answer in valid JSON format as follows:
{
  "rationale": "Your extensive planning, step-by-step reasoning, and assertions...",
  "answer": "The final answer (e.g. 'A', 'B', '42', 'summary')"
}
```

PRBench Finance

```
You are a Chief Financial Officer and Expert Financial Analyst.
The user will provide a Finance Question or Context.
Your goal is to provide the most ACCURATE and PRECISE answer possible.

Guidelines:
1. PRECISION: Perform all calculations with high precision. Double-check your arithmetic.
2. FIRST PRINCIPLES: Derive answers from fundamental financial concepts and regulations.
3. CONSTRAINTS: Pay close attention to every constraint and detail in the prompt.
4. COMPLETENESS: Ensure the final answer addresses the specific question asked without
    ambiguity.
5. REASONING: Briefly explain your steps to ensure correctness before stating the final
    answer.

Output JSON:
{
  "answer": "The final answer..."
}
```

ARC AGI 2

```
I will provide you with several input and output matrices about 2D grids. You need to
    summarize the grid-changing rule from it.

Step 1: Rule Extraction First, find the matrix-changing rule from the examples.
Step 2: Rule Verification Check, the correctness of the rule based on the examples. If the
    rule is correct, apply it to the new input. Otherwise, summarize a new rule and apply it
     to the new input.
Step 3: Apply the finalized rule to the new input. Put the output matrix within \boxed{}.

Example input 1 [Matrix Data] [Image of Matrix]
Example output 1 [Matrix Data] [Image of Matrix]
... [Repeats for all training examples] ...

New Input [Matrix Data] [Image of Matrix]
```

A.7 Self-Consistency Implementation

For the Self-Consistency (SC) baselines, we utilized the same generation prompts as the standard baseline but increased the sampling budget to $N = 8$ and $N = 16$ independent responses per problem.

For tasks with structured outputs (HLE and ARC-AGI), the final prediction was selected via standard majority voting based on exact string matching. However, as PRBench Finance requires complex, free-

form reasoning, exact matching is insufficient for determining consensus. To address this, we employed an additional LLM-based aggregation step to identify the semantically most common answer among the generated samples. The specific prompt used for this consensus extraction is provided below:

Prompt for Majority vote

```
You are an expert consensus judge.
You will be provided with a Question/Context, and 16 different candidate answers generated by
     an AI.
Your task is to analyze these candidate answers, determine the "majority vote" or most common
     distinct answer, and output the final consensus answer.

Think carefully about which answer is truly the most common conceptually, accounting for
    minor formatting differences or rounding.
Provide your reasoning, and then output your final majority consensus answer in the specified
     JSON format.

Output JSON:
{
  "reasoning": "Explanation of the voting...",
  "answer": "The final consensus majority answer..."
}
```

B Multi-Agent Consensus (CoScientist) Ablation

To investigate whether inter-agent collaboration could bridge the The Verification Bottleneck, we implemented a "CoScientist" consensus pipeline Gottweis et al. (2025). This architecture decomposed the selection process into five specialized roles: *Proximity* (clustering), *Ranking* (tournament comparison), *Reflection* (peer review), *Evolution* (refinement), and *Meta-Review* (final polish).

We evaluated this pipeline by feeding the raw PoTRE candidate outputs directly into the consensus agent. Empirical results indicate that this added complexity did not yield improvements over existing Synthesizer, and in some cases degraded performance:

- **Humanity's Last Exam (HLE):** The consensus pipeline achieved an accuracy of **39.60%** (990/2500), under performing the standard PoTRE 3.1 Pro result (49.92%).
- **PRBench Finance:** The pipeline attained an Average Clipped Score of **0.3185** (vs. 0.3486 for PoTRE).
- **ARC-AGI-2:** Accuracy dropped to **19.16%** (23/120), significantly lower than the PoTRE performance (38.30%).

These findings suggest that while iterative debate is intuitively appealing, it may introduce "consensus collapse," where agents converge on plausible-sounding but incorrect answers, particularly in abstract reasoning tasks. For transparency, the exact system prompts used for each role are provided below.

B.1 Prompts for HLE and PRBench Finance

Proximity Agent (Data Clustering)

```
system_instruction = "You are a Data Clustering Agent."
```

```
prompt = f"""Analyze the following candidate answers. Group IDs that represent the
    SUBSTANTIALLY SAME answer or logic.
Output a JSON list of lists, where each inner list contains IDs of identical answers.
Example: [[0, 2], [1], [3, 4]]

Candidates:
{candidates_text}
"""
```

Ranking Agent (Tournament logic)

```
system_instruction = "You are a Tournament Judge. Pick the best answer."

prompt = f"""Compare Option A and Option B for the following problem.
Which is more accurate, rigorous, and complete?
Output ONLY 'A' or 'B'.

Problem:
{self.context}

Option A:
{ans_a[1]}

Option B:
{ans_b[1]}
"""
```

Reflection Agent (Peer Reviewer)

```
system_instruction = "You are a Scientific Peer Reviewer."

prompt = f"""Review the following solution to the problem below.
Identify:
1. Correctness errors
2. Missing details or derivations
3. Logical gaps or weak reasoning
Be extremely critical and rigorous.

Problem Context:
{self.context}

Proposed Solution:
{candidate_answer}
"""
```

Evolution Agent (Refinement)

```
system_instruction = "You are a Research Evolution Agent."

prompt = f"""You are the Evolution Agent.
Refine the following 'Best Answer' based on the 'Critique'.
Synthesize existing knowledge, fix errors, and clarify concepts.
```

```
Ensure the final answer is complete and solves the original problem perfectly.

Problem:
{self.context}

Best Answer So Far:
{best_candidate[1]}

Critique from Peer Review:
{critique}

Task: Rewrite the answer to be perfect.
"""
```

Meta-Review Agent (Final Polish)

```
system_instruction = "You are a Meta-Review Editor."

prompt = f"""Finalize this answer.
Ensure the tone is professional, authoritative, and precise.
Output ONLY the final answer content. Do not include "Here is the answer" or similar
    preambles.

Draft:
{final_draft}
"""
```

B.2 Prompts for ARC AGI 2

Proximity Agent (Data Clustering)

```
Analyze the following candidate answers. Group IDs that represent the SUBSTANTIALLY SAME
    answer, logic, or grid structure.
Output a JSON list of lists. Example: [[0, 2], [1], [3]]

Candidates:
[List of candidate IDs and their answers]
```

Ranking Agent (Tournament Logic)

```
Compare Option A and Option B for the following task.
Which is more accurate, correct, and follows the pattern?
For ARC/Grid tasks, check which output accurately follows the transformation rules seen in
    training for ALL test inputs.
Output ONLY 'A' or 'B'.

Task Context:
[Task Context string containing training and test grids]

Option A:
[Candidate A string]
```

```
Option B:
[Candidate B string]
```

Reflection Agent (Peer Reviewer

```
Review the following solution.
Identify:
1. Pattern violations (if ARC/Grid task)
2. Logical gaps
3. Correctness errors
Be extremely critical.

Context:
[Task Context string containing training and test grids]

Proposed Solution:
[The winning candidate answer from the Ranking Agent]
```

Evolution Agent (Refinement)

```
You are the Evolution Agent.
Refine the 'Best Answer' based on the 'Critique'.
If this is an ARC Grid task, ensure the final output contains perfectly valid grids for ALL
    test inputs and follows the training pattern.
Output the FINAL, CORRECTED answer.

Context:
[Task Context string containing training and test grids]

Best Answer So Far:
[The winning candidate answer from the Ranking Agent]

Critique:
[The critique generated by the Reflection Agent]
```

Meta-Review Agent (Final Polish)

```
Finalize this answer.
Output ONLY the final answer content as a SINGLE valid JSON array containing ALL the test
    output grids.
For example, if there are two test inputs, output:
[
  [[0, 0], [0, 0]],
  [[1, 1], [1, 1]]
]
Do NOT include any explanations, markdown formatting, or multiple code blocks. Return ONLY
    the JSON array.

Draft:
[The refined final draft from the Evolution agent]
```

Algorithm 2 Adversarial Refinement Agent

Require: Task Context $\mathcal{T}$ (text, images, or grid pairs), Proposer Agent $\mathcal{A}_P$, Verifier Agent $\mathcal{A}_V$, Max Debate Turns N_{max}
Ensure: Final Consensus Solution S_{final} (Code or JSON structure)

```
% Step 1: Initialization & Initial Proposal
S_cand ← A_P.GenerateProposal(T)          ▷ Initial generation (hypothesis, logic, or code)
turn ← 0
% Step 2: Debate & Verification Loop
while turn < N_max do
    Critique, Status ← A_V.Evaluate(S_cand, T)          ▷ Verifier analyzes the candidate
    if Status = APPROVED then
        break                                           ▷ Consensus reached
    end if
    Feedback ← FORMATREBUTTAL(Critique)
    S_cand ← A_P.RefineProposal(S_cand, Feedback)       ▷ Proposer incorporates critique
    turn ← turn + 1
end while
% Step 3: Final Output
S_final ← S_cand
return S_final                ▷ Returns agreed solution, or best-effort if max turns reached
```

Table 13: **Claude-4.5-Sonnet Thinking Results.** Performance evaluation of the PoTRE framework using the Claude-4.5-Sonnet Thinking model across three distinct benchmarks. The results demonstrate consistent improvements over both baseline and Self-Consistency (SC).

	Baseline Evaluation			PoTRE Evaluation	
Benchmark	Prior Work Prompts	SC ($N = 8$)	SC ($N = 16$)	Score	Improv.[‡]
Humanity's Last Exam	12.08%	12.39%	12.61%	**15.20**%	+3.12
ARC-AGI 2	5.00%	5.00%	6.66%	**10.83**%	+5.83
PRBench Finance Hard	0.2625	0.2222	0.2343	**0.5196**	**+0.2571**

[‡]Improvement calculated vs. Baseline (Prior work prompts).

C Algorithms for each Sub-Agent

All algorithms for, Adversarial Refinement agent, Hierarchical Strategic Planning Agent, Spectrum Search Agent and Direct Chain Agent are provided as Algorithm 2, Algorithm 3, Algorithm 4 and Algorithm 5 respectively.

D Experiments with Claude Sonnet 4.5 Thinking

To verify that PoTRE's architectural gains are fundamentally model-agnostic and not overfit to the Gemini family, we extended our evaluation to Claude-4.5-Sonnet Thinking. The comprehensive results across all three benchmarks are detailed in Table 13. On the PRBench Finance - Hard subset, this model achieved an average clipped score of **0.5196**, surpassing the baseline of 0.2713 and nearly doubling the margin by **0.2571** points. These results represent a new state-of-the-art on this benchmark[4].

Note on ARC-AGI-2 Performance: The official task page[5] indicates that Claude-4.5-Sonnet Thinking (across 8K, 16K, and 32K configurations) failed all tasks within the public evaluation sets. This baseline deficiency

[4]Official Leaderboard: `https://scale.com/leaderboard/prbench-finance`
[5]ARC-AGI-2 Tasks: `https://arcprize.org/tasks/?dataset=arc-agi-2&model=claude-sonnet-4-5-20250929-thinking-32k`

Table 14: **Component analysis on HLE.** Performance breakdown of individual reasoning agents across 2500 tasks. The "Exclusive" column indicates the number of tasks solved uniquely by that specific agent. The combined Oracle upper bound (where at least one agent produced the correct answer) reached 592 solved tasks (23.7%).

Sub-Agents	Solves	Exclusive	Accuracy
Adversarial Refinement Agent	**319**	**92**	**12.76%**
Hierarchical Strategic Planning Agent	283	66	11.32%
Direct Chain Agent (CoT)	283	60	11.32%
Spectrum Search Agent	287	57	11.48%
Oracle Upper Bound	**592**	–	**23.7%**

highlights the model's inherent struggle with this specific spatial reasoning benchmark, effectively contextualizing the low absolute accuracy observed in our results.

D.1 Ablation Study: Contribution of Individual agents

We conducted same ablation study we've performed for other models to understand the contributions of each sub-agent.

D.1.1 Humanity's Last Exam (HLE)

To validate the necessity of a diverse multi-agent ensemble, we conducted a component ablation on the HLE dataset, tracking both total and uniquely solved tasks across all 2500 problems (Table 14). While the Adversarial Refinement Agent achieved the highest independent performance (319 solved, 12.76% accuracy), the most critical finding lies in the "Exclusive" column. Every distinct reasoning topology successfully resolved a substantial number of tasks that all other agents failed to answer, ranging from 57 to 92 uniquely solved problems per agent. Because these orthogonal reasoning paths effectively cover each other's cognitive blind spots, the theoretical Oracle upper bound—representing tasks where at least one agent produced the correct answer—reaches 592 solved tasks (23.7%). This nearly doubles the highest single-agent baseline, empirically demonstrating that cognitive heterogeneity is essential for maximizing performance on highly constrained, complex reasoning benchmarks.

D.1.2 PRBench Finanace Hard

To further understand the topological preferences of different frontier models, we conducted a component ablation on the PRBench Finance Hard subset (Table 15). The analysis reveals a distinct architectural preference: Claude-4.5-Sonnet Thinking strongly favors the Hierarchical Strategic Planning Agent as its primary reasoning engine, achieving a leading individual Average Clipped score of 0.4038. However, the true strength of the PoTRE framework emerges at the synthesis agent. By integrating the orthogonal reasoning trajectories generated by the parallel sub-agents, the final synthesis unlocks a massive domain-specific performance gain for Claude, elevating its Average Clipped score from 0.4038 to a state-of-the-art 0.5196.

D.1.3 ARC-AGI 2

Turning to spatial reasoning on ARC-AGI-2 (Table 16), this model demonstrates a distinct reliance on the Spectrum Search Agent, which leads the individual sub-agents with 9 solved tasks. The PoTRE final synthesis successfully integrates these orthogonal reasoning paths, elevating the overall performance to 13

Table 15: **Component Analysis on PRBench Finance (Hard).** Performance breakdown of individual reasoning agents compared to the final PoTRE synthesis. While the Hierarchical Strategic Planning Agent serves as the strongest individual topology, the final synthesis provides a massive lift, confirming the value of multi-agent aggregation.

Sub-Agents	Avg Norm	Avg Clip
Adversarial Refinement Agent	0.4095	0.3613
Hierarchical Strategic Planning Agent	**0.4507**	**0.4038**
Spectrum Search Agent	0.4346	0.3882
Direct Chain Agent	0.4069	0.3602
PoTRE Final Synthesis	**0.5565**	**0.5196**

Table 16: **Component Analysis on ARC-AGI-2.** Performance breakdown of individual reasoning agents compared to the PoTRE final synthesis. The Spectrum Search Agent serves as the primary reasoning engine, while the synthesis agent further elevates the final score toward the Oracle upper bound.

Sub-Agents	Solved	Exclusive
Adversarial Refinement Agent	7	1
Hierarchical Strategic Planning Agent	3	0
Spectrum Search Agent	**9**	**7**
Direct Chain Agent	3	1
Oracle Upper Bound	**15**	–
PoTRE Final Synthesis	**13**	–

solved tasks. This demonstrates a clear improvement over the best single-agent baseline, though it remains below the theoretical Oracle upper bound of 15.

Summary of Ablation Across three highly distinct domains—scientific knowledge (HLE), complex financial reasoning (PRBench), and strict spatial logic (ARC-AGI-2)—our ablation studies consistently demonstrate that the PoTRE framework is fundamentally greater than the sum of its individual components. Rather than relying on a single dominant reasoning topology, the architecture actively exploits cognitive heterogeneity. On HLE, this diversity proved essential, as orthogonal agents solved distinct, mutually exclusive subsets of problems, effectively doubling the best single-agent baseline. Conversely, on PRBench and ARC-AGI-2, while the base models exhibited clear topological preferences for specific domains (favoring Hierarchical Planning for finance and Spectrum Search for spatial tasks), the final synthesis agent consistently extracted marginal value from the parallel agents. By successfully integrating these diverse reasoning trajectories, PoTRE systematically narrows the gap toward theoretical Oracle upper bounds, confirming that structured, multi-agent aggregation is a robust and domain-agnostic mechanism for scaling inference-time compute.

E Experiments with Deepseek-V3.2-maas

To further establish the model-agnostic versatility of the PoTRE framework, we expanded our evaluation to include the DeepSeek model. As given in table 17, the PoTRE-enhanced DeepSeek achieved an accuracy of 20.20% on the Humanity's Last Exam (HLE) text-only subset. Furthermore, on the complex financial reasoning tasks of the PRBench Finance - Hard subset, the model delivered an average clipped score of 0.3499. These results reinforce that our multi-agent synthesis effectively extracts structural reasoning gains

Table 17: **DeepSeek Results.** Performance evaluation of the PoTRE framework using the DeepSeek model across distinct reasoning benchmarks. The results highlight the multi-agent synthesis gains compared to standard baselines and Self-Consistency (SC).

	Baseline Evaluation			PoTRE Evaluation	
Benchmark	Prior Work Prompts	SC ($N = 8$)	SC ($N = 16$)	Score	Improv.[‡]
Humanity's Last Exam	18.16%	19.46%	19.83%	**20.20**%	+2.04
PRBench Finance Hard	0.2299	0.0717	0.0991	**0.3499**	**+12.00**

[‡]Improvement calculated vs. Baseline (Prior work prompts).

Table 18: **Component Analysis of DeepSeek on HLE (Text-Only).** Performance breakdown of individual reasoning agents across 2158 tasks. The "Exclusive" column indicates the number of tasks solved uniquely by that specific agent. The combined Oracle upper bound (where at least one agent produced the correct answer) reaches 696 solved tasks (32.3%).

Sub-Agents	Solves	Exclusive	Accuracy
Spectrum Search Agent	**441**	**128**	**20.4%** (441/2158)
Hierarchical Strategic Planning Agent	335	66	15.5% (335/2158)
Direct Chain Agent	332	66	15.4% (332/2158)
Adversarial Refinement Agent	245	64	11.4% (245/2158)
Oracle Upper Bound	**696**	–	**32.3%**

across diverse frontier architectures, proving the framework's viability extends well beyond the Gemini and Claude models.

E.1 Ablation Study: Contribution of Individual agents

We conducted same ablation study we've performed for other models to understand the contributions of each sub-agent.

E.1.1 Humanity's Last Exam (Text-Only)

To unpack the structural advantages of the PoTRE framework when applied to the DeepSeek architecture, we conducted a component ablation on the HLE text-only subset (Table 18). The Spectrum Search Agent emerges as the dominant individual topology for this model, independently solving 441 tasks (20.4% accuracy) and contributing a massive 128 exclusive solves. The ensemble's cognitive heterogeneity remains robust, with the remaining agents successfully resolving between 64 and 66 exclusive tasks each, expanding the theoretical oracle upper bound to 696 solved tasks (32.3%). However, the actual PoTRE synthesis achieves only 20.20% accuracy, slightly trailing the best single-agent baseline. This substantial gap between the realized performance and the oracle bound exposes a critical limitation: while the parallel agents successfully generate the correct reasoning path, the final synthesis agent struggles to consistently filter out the noise from incorrect agents. Rather than extracting these unique correct answers, the synthesis bottleneck occasionally degrades the optimal yield, highlighting a primary area for future routing and consensus optimization.

Table 19: **Component Analysis of DeepSeek on PRBench Finance (Hard).** Performance breakdown of individual reasoning agents compared to the final PoTRE synthesis. The synthesis agent successfully integrates diverse reasoning paths to achieve a final Average Clipped score of 0.3499, significantly outperforming the best individual agent.

Sub-Agents	Avg Norm	Avg Clip
Spectrum Search Agent	**0.3577**	**0.3053**
Adversarial Refinement Agent	0.3509	0.2993
Direct Chain Agent	0.3277	0.2730
Hierarchical Strategic Planning Agent	0.3260	0.2701
PoTRE Final Synthesis	**0.3987**	**0.3499**

E.2 PRBench Finance (Hard)

To evaluate DeepSeek's performance on complex financial reasoning, we conducted a component ablation on the PRBench Finance Hard subset (Table 19). The DeepSeek model strongly favors the Spectrum Search Agent, which achieves a leading individual average clipped score of 0.3053. Notably, unlike the synthesis bottleneck observed in the HLE text-only evaluation, the PoTRE framework proves highly effective in this domain. By successfully synthesizing the orthogonal reasoning trajectories generated by the parallel sub-agents, the architecture elevates the final Average Clipped score to 0.3499. This substantial gain over the best single-agent baseline confirms that structured, multi-agent aggregation effectively extracts and amplifies the underlying model's financial reasoning capabilities.

Algorithm 3 Hierarchical Strategic Planning Agent

Require: Task Context $\mathcal{T}$, Agent Ensemble $\{\mathcal{A}_{Arch}, \mathcal{A}_{Eng}, \mathcal{A}_{Sup}, \mathcal{A}_{Meta}\}$, Max Turns T_{max}, Max Debug Retries D_{max}
Ensure: Final Answer A_{final}

$L_{history} \leftarrow [\,]$ ▷ Log of execution trajectories
$P_{Arch} \leftarrow$ Base System Prompt
for $t = 1$ **to** T_{max} **do**
  % Phase 1: Meta-Strategic Intervention
  if $\textsc{ShouldIntervene}(t, L_{history})$ **then**
    $Review \leftarrow \mathcal{A}_{Meta}.\text{AnalyzeTrajectory}(L_{history}, \mathcal{T})$
    $P_{Arch} \leftarrow \textsc{InjectHeuristic}(P_{Arch}, Review)$ ▷ Force strategic pivot
  end if
  % Phase 2: Hypothesis Generation
  $Plan \leftarrow \mathcal{A}_{Arch}.\text{GeneratePlan}(\mathcal{T}, P_{Arch})$
  % Phase 3: Concept Verification (For Formal/Strict Tasks)
  if $\mathcal{T}$ requires strict logical constraints **then**
    $Valid, Feedback \leftarrow \mathcal{A}_{Sup}.\text{EvaluateLogic}(Plan)$
    if not $Valid$ **then**
      $P_{Arch} \leftarrow \text{Update}(P_{Arch}, Feedback)$ ▷ Reject flawed logic immediately
      continue
    end if
  end if
  % Phase 4: Implementation
  $Draft \leftarrow \mathcal{A}_{Eng}.\text{GenerateDraft}(\mathcal{T}, Plan)$
  % Phase 5: Domain-Adaptive Validation & Engineer Auto-Debugging
  if $\mathcal{T}$ requires environment execution (e.g., ARC) **then**
    $d \leftarrow 0$, $Success \leftarrow$ False
    while $d < D_{max}$ **and not** $Success$ **do** ▷ Internal Engineer Repair Loop
      $Valid, ErrorMsg \leftarrow \textsc{ExecuteAndReview}(Draft, \mathcal{T})$
      if $Valid$ **then**
        $Success \leftarrow$ True
      else
        $Draft \leftarrow \mathcal{A}_{Eng}.\text{RefineDraft}(Draft, ErrorMsg)$ ▷ Pass tracebacks to Engineer
        $d \leftarrow d + 1$
      end if
    end while
    if not $Success$ **then**
      $P_{Arch} \leftarrow \text{Update}(P_{Arch}, \text{“Execution Failed: ”} + ErrorMsg)$ ▷ Propagate to Architect
      continue
    end if
    $A_{final} \leftarrow Draft$
  else ▷ Factual / reasoning tasks (e.g., HLE)
    $A_{final}, Rationale \leftarrow \mathcal{A}_{Sup}.\text{ReviewAndSynthesize}(Draft, \mathcal{T}, Plan)$
    $Success \leftarrow$ True
  end if
  $L_{history}.\text{append}(\{Plan, Draft, Success\})$
  if $Success$ **then**
    break ▷ Task solved, exit macro-loop
  end if
end for
return A_{final}

Algorithm 4 Spectrum Search Agent

Require: Task Context $\mathcal{T}$, Number of generative agents N
Ensure: Optimal verified solution s^*

% Initialization
$\mathcal{C} \leftarrow \emptyset$ ▷ Candidate solution pool
% Phase 1: High-Concurrency Hypothesis Generation
for $i = 1, \ldots, N$ **in parallel do**
  $\tau_i \leftarrow \text{ScaleTemperature}(i, N)$ ▷ Linearly scale temperature $T \in [0.2, 1.0]$ for diversity
  $c_i \leftarrow \text{GeneratorAgent}_\theta(\text{prompt} = \mathcal{T}, \text{temp} = \tau_i)$
  if $\mathcal{T}$ requires Code Execution (e.g., ARC) **then**
    $c_i \leftarrow \text{RefinementLoop}(c_i, \mathcal{T}, k_{\max} = 4)$ ▷ Iterative self-critique prior to execution
  end if
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_i\}$
end for
% Phase 2: Domain-Specific Verification & Consensus
if $\mathcal{T}$ requires Code Execution (e.g., ARC) **then**
  % Execution-Based Verification
  $\mathcal{D}_{\text{train}} \leftarrow \text{ExtractConstraints}(\mathcal{T})$
  $\mathcal{C}_{\text{valid}} \leftarrow \{c \in \mathcal{C} \mid c(x) = y, \forall (x, y) \in \mathcal{D}_{\text{train}}\}$
  if $\mathcal{C}_{\text{valid}} = \emptyset$ **then**
    $\mathcal{C}_{\text{valid}} \leftarrow \text{ExecutionFeedbackRepair}(\mathcal{C}, \mathcal{D}_{\text{train}})$ ▷ Traceback-guided repair
  end if
  $Y_{\text{consensus}} \leftarrow \text{MajorityVote}(\{c(\mathcal{T}_{\text{test}}) \mid c \in \mathcal{C}_{\text{valid}}\})$
  $s^* \leftarrow \arg\min_{c \in \mathcal{C}_{\text{valid}}, c(\mathcal{T}_{\text{test}}) = Y_{\text{consensus}}} \text{Complexity}(c)$
else
  % Semantic/Factual Verification (e.g., HLE)
  $\mathcal{P}_{\text{eval}} \leftarrow \mathcal{T} \cup \text{Serialize}(\mathcal{C})$
  $idx^*, \text{rationale}^* \leftarrow \text{ExpertJudgeAgent}_\psi(\text{prompt} = \mathcal{P}_{\text{eval}}, \text{temp} = 0.0)$
  $s^* \leftarrow \mathcal{C}[idx^*]$ ▷ Select highest fidelity solution candidate
end if
return s^*

Algorithm 5 Direct Chain Agent

Require: Task instance $\mathcal{T}$, Target model $\mathcal{M}$, Max API retries R_{max}
Ensure: Structured output O (e.g., Parsed JSON or Executable Code)

```
% Step 1: Initialization & Configuration
P ← ConstructPrompt(T)                    ▷ Domain-specific prompt, incl. images/text/examples
config ← {temperature : τ}
attempt ← 0
wait_time ← W_init                        ▷ Initial delay penalty
% Step 2: Robust Execution Loop
while attempt < R_max do
    try
        Response ← GenerateContent(M, P, config)
        if Response is null or Response.text is empty then
            raise ResponseContentError
        end if
    % Step 3: Parsing & Validation
        O ← RobustParse(Response.text)    ▷ Regex extraction & fallback logic
        if O is valid and meets structural constraints then
            return O
        else
            raise StructuralParsingError
        end if
    catch APIError E                      ▷ e.g., 429 Rate Limit, 503 Service Unavailable
    % Step 4: Exponential Backoff with Jitter
        jitter ← Uniform(0.8, 1.2)        ▷ Prevent synchronized retry cascades
        delay ← min(W_max, wait_time × jitter)
        Sleep(delay)
        wait_time ← min(W_max, wait_time × α)   ▷ Exponential multiplier, e.g., α = 1.5 or 2.0
        attempt ← attempt + 1
    end try
end while
raise ExceededMaxRetriesError
```