# Formula-One Prompting: A Composable Equation-First Prefix for Applied Mathematics

Anonymous ACL submission


## Abstract

This paper introduces **Formula Prompting** (FP) and **Formula-One Prompting** (F-1), two single-call methods that elicit governing equations before solving applied-math problems. Chain-of-Thought (CoT) and Program-of-Thought (PoT) prompting improve mathematical reasoning by eliciting reasoning traces or code-like structures learned during pretraining. This suggests a diagnostic question: which useful pretraining patterns remain under-elicited? Using INFINI-GRAM-MINI, we scan 81.7 trillion pretraining tokens and find that, in curated corpora such as DataComp-LM, equation-centered language appears 121× more often than code and 3.79× more often than step-by-step narration, yet standard prompting methods do not explicitly elicit equation formulation. FP asks the model to formalize a problem's governing equations before solving; F-1 extends FP with a composable Phase 2 that selects Direct, CoT, or PoT-style solving in the same call. Across five reasoning models and four applied-math benchmarks (finance, physics, cryptography, competition math), F-1 outperforms CoT by 5.76 pp and PoT by 8.42 pp on average, with the largest gain of 13.30 pp on FinanceMath, while topping the accuracy-token efficiency frontier at only 68 prompt tokens of overhead. Ablations show that the equation-formalization prefix, rather than the Phase-2 strategy menu, drives most of the gains: adding CoT or PoT after the prefix provides no further improvement, and 73.3% of remaining errors occur despite a correct Phase-1 equation.


## 1 Introduction

Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) and Program-of-Thought (PoT) (Chen et al., 2023b) show that prompting can improve mathematical reasoning by eliciting formats

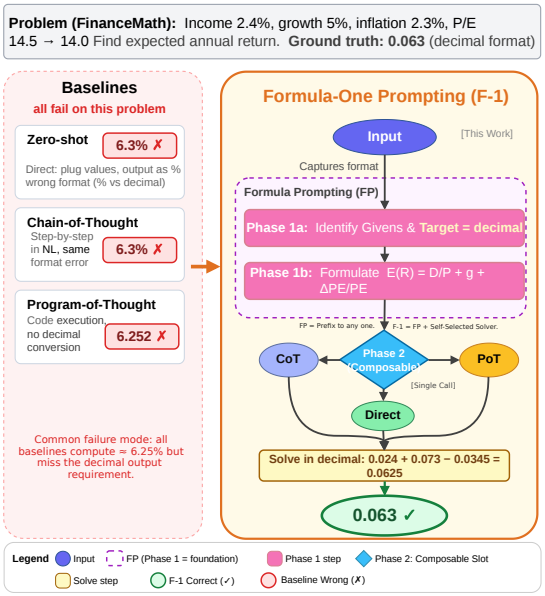


Figure 1: F-1 compared to other prompting methods on a FinanceMath problem (about Grinold-Kroner; ground truth is 0.063). Baselines compute ≈6.25% but miss the decimal-format requirement; F-1's Phase 1 captures both equation and format, and the Phase 2 chooses any solving strategy.

present in pretraining (Prystawski et al., 2023; McCoy et al., 2024), including step-by-step explanations and code. We later quantify this prevalence directly (Section 3). Applied mathematics, e.g., problems in finance, physics, and cryptography, often depend on domain-specific *equations*, including compound interest formulas, Newton's laws, and security game definitions (Wang et al., 2024; Xu et al., 2025b). Large math corpora also contain substantial LaTeX and equation-heavy content (Lewkowycz et al., 2022; Paster et al., 2023; Shao et al., 2024). Therefore, we investigate the effectiveness of *explicitly* eliciting a model to formulate equations before solving a problem. This contrasts with existing prompting approaches such as CoT and PoT, which may generate equations implicitly

during the solving process, but do not consistently require them.

To measure this opportunity, we run an infini-gram-mini scan over 81.7T tokens (Section 3). In DCLM (Li et al., 2025), equation-language, including LaTeX typography patterns, occurs 121× more often than code and 3.79× more often than step-by-step narration. Following evidence that pretraining frequency predicts downstream task performance (Razeghi et al., 2022; Prystawski et al., 2023; McCoy et al., 2024), we use this prevalence as a diagnostic signal: equation-language patterns may be an *under-used promptable pattern*. Existing prompting methods target other forms of mathematical reasoning–CoT elicits natural-language reasoning, PoT elicits code generation, and Plan-and-Solve (Wang et al., 2023a) elicits planning–but they do not explicitly ask the model to formulate the governing equations central to problem solving.

To understand the effectiveness of explicit formulation-first instruction, which we call **Formula Prompting (FP)**, we compare FP against a zero-shot baseline and run four analyses to study what the prefix contributes: per-problem flips against a fixed CoT solver (Section 5.3), output-pattern composability across solvers (Section 5.4), variant accuracy at matched reasoning budget (Section 5.5), and a failure-cause classification showing that 73.3% of remaining failures have a correct Phase-1 equation but fail in the downstream solver (Section 5.6). Motivated by these analyses, we extend FP into **Formula-One Prompting (F-1)** (Figure 1) with a composable second phase that selects Direct, CoT, or PoT solving within a single inference call, preserving equation formalization as the shared intervention. F-1 outperforms CoT by 5.76 pp and PoT by 8.42 pp on average. While Formulate-and-Solve prompting (Kao et al., 2024) also formulates equations, F-1 separates equation formulation from the composable selection of a solving strategy (Table 5). Our contributions are:

- **Formula Prompting (FP)** A novel equation-first prompting approach. Our experiments show that explicitly eliciting equation formulation before solving is effective compared with a zero-shot baseline in most included model-benchmark combinations (Section 5).
- **Formula-One Prompting (F-1)** A composable extension of FP that adds a second phase selecting Direct, CoT, or PoT solving within a single call. Experiments show that F-1 outperforms CoT and PoT prompting in most cases and achieves higher token efficiency than baselines (Section 6).
- **Pretraining-prevalence diagnostic** An infini-gram-mini scan over 81.7T pretraining tokens. We measure the relative prevalence of equation-language, code, and step-by-step narration patterns, supporting the motivation for explicit equation elicitation (Section 3).

## 2 Related Work

### 2.1 Single-Call Prompt Engineering

**Chain-of-Thought** (Wei et al., 2022; Kojima et al., 2022) improves LLM reasoning by eliciting step-by-step natural-language explanations, but its arithmetic and derivation steps can accumulate errors (Chen et al., 2023b; Sprague et al., 2025). **Program-of-Thought** (Chen et al., 2023b) reduces computation errors through executable code, but is most useful when problems are programmatically expressible; CoT and PoT have been shown to be complementary across reasoning tasks (Liu et al., 2023). **Plan-and-Solve** (Wang et al., 2023a) adds explicit planning, but structures the execution process rather than the problem representation. None of these methods explicitly formalize governing equations before solving. For applied-math domains where governing equations are identifiable, such as finance, physics, and cryptography (Wang et al., 2024; Xu et al., 2025b), this leaves the symbolic structure implicit during reasoning. For tasks without such structure, explicit formalization is not appropriate; our negative-control experiment (Section 6.7) shows it can in fact hurt.

Beyond single-call prompting, search and routing methods such as **ToT** (Yao et al., 2023), **GoT** (Besta et al., 2024), **XoT** (Liu et al., 2023), and **Adaptive-Solver** (Zhou et al., 2024) explore multiple reasoning paths or route each problem to a chosen solver, at substantially higher computational cost (3 to 100+ model invocations per problem). F-1 targets the single-call regime and is complementary to these search-based approaches (Table 5).

Prior work explores equation formalization: MathPrompter (Imani et al., 2023) uses generic algebraic templates, He-Yueya et al. (2023) rely on multi-call SymPy solving, and Formulate-and-Solve (F&S) (Kao et al., 2024) prompts equation-first solutions for algebra word problems. F-1 differs from this line of work along three axes. **(1)**

**Single-call, no external tools.** F-1 needs no symbolic solver or external router (unlike He-Yueya et al., 2023). **(2) Composable Phase 2.** F-1 offers a Direct, CoT, or PoT menu within the same call (unlike F&S's fixed algebraic solve). **(3) Applied-math focus.** Prior equation-based methods target generic algebra; F-1 targets applied-math domains such as finance, physics, and cryptography, whose governing equations are domain-canonical (e.g., compound interest, Newton's laws, security games) rather than ad-hoc per-problem expressions. This makes formalization a structural rather than incidental intervention. Table 5 summarizes design differences; Table 4 reports a head-to-head accuracy comparison with F&S in the main body, with full per-problem cross-tab in Section Q.

## 3 Pretraining Pattern Diagnostic

**Objective.** We test whether the empirical effectiveness of equation-first prompting is partly predicted by the prevalence of equation-language in pretraining, consistent with prior findings that pretraining frequency correlates with downstream task performance (Razeghi et al., 2022; Prystawski et al., 2023; McCoy et al., 2024).

**Method.** We query the Infini-gram-mini API (Xu et al., 2025a) for n-gram occurrences of method-associated patterns across eight pretraining corpora ($\sim$81.7T tokens): DCLM-baseline (16.7T curated tokens) and seven Common Crawl snapshots. For each method (CoT, PoT, FP) we instantiate two pattern classes: **prompt phrases** taken verbatim from its prompt template (step-by-step language for CoT, code instructions for PoT, equation-formalization language plus LaTeX environments for FP), and **solving vocabulary** the method tends to produce in outputs (math notation including LaTeX for FP, code imports for PoT). Per-pattern occurrence ratios on actual model outputs validate method-specificity: associated patterns appear at 10$\times$ or higher frequency in their own outputs than in others'. Full pattern lists and validation in Section L.

**Hypothesis.** If prevalence predicts elicitability, FP-associated patterns should be at least as prevalent as CoT-associated patterns in curated corpora, and substantially more prevalent than PoT-associated patterns. The PoT gap would predict FP's largest lift over PoT in particular.

**Results.** On DCLM (curated), FP-associated patterns occur **121**$\times$ more often than PoT-associated patterns (69.6M vs. 0.6M occurrences) and **3.79**$\times$ more often than CoT-associated patterns (69.6M vs. 18.4M). The single LaTeX command `\frac{` alone (23.9M) exceeds all PoT patterns combined. The FP/PoT ratio remains 19–61$\times$ across the seven raw Common Crawl snapshots. To rule out that a few equation-heavy documents drive the count, we re-tabulated by unique source document rather than raw occurrences (Section S): each method averages $\approx$1.0 occurrence per containing document, so the ranking is not a concentration artifact. See Figure 5 in Section L for the full breakdown.

**Interpretation and scope.** The diagnostic provides *correlational* evidence: pattern abundance in pretraining is consistent with FP's empirical lift reported in later sections, but the mechanism behind this lift is out of scope. We use the analysis to motivate the design of FP and to bound its applicable domain; mechanistic claims about how models represent equations or why equation-first prompting changes their behavior are left for future work.

## 4 Experimental Setup

All experiments in this paper follow the setup described here, unless explicitly stated otherwise.

**Datasets.** We evaluate all prompting approaches on four applied-mathematics benchmarks where formulation-first reasoning may help, spanning physics problems that invoke Newton's laws or thermodynamic equations, finance problems requiring compound-interest or pricing models, and cryptography problems requiring security-game formalizations. The included benchmarks are listed in Table 1.

| Benchmark | Domain | N | Eval. |
|---|---|---|---|
| IMO-Bench (Luong et al., 2025) | Comp. Math | 460 | Rule/LLM |
| OlympiadBench (He et al., 2024) | Math + Physics | 1,438 | Rule/LLM |
| FinanceMath (Zhao et al., 2024) | Applied Finance | 200 | Rule |
| AICrypto (Wang et al., 2025) | Cryptography | 18 | LLM |

Table 1: Evaluation benchmarks (total: 2,116 problems). Rule = regex-based answer matching; LLM = LLM-as-Judge.

We select benchmark samples using three criteria: (1) text-only, open-answer format requiring mathematical derivation; (2) unsaturated by frontier models ($< 95\%$ accuracy); and (3) equation-driven problems with identifiable governing formulas. For AICrypto, we use only proof problems, excluding MCQs and CTF challenges that primarily test recall; for OlympiadBench, we use the English

text-only subset; and for FinanceMath, we use the public splits with ground-truth answers. OlympiadBench also includes both competition mathematics and applied physics, enabling controlled cross-domain comparison.

**Evaluation Protocol.** We use temperature 0 with default inference hyperparameters, evaluating each question with a single greedy output. Definitive-answer problems are scored by regex-based extraction with numerical tolerance $\epsilon = 10^{-6}$; proof-based problems use domain-appropriate LLM-as-Judge evaluation (Zheng et al., 2023). Details and judge prompts appear in Section F.

**Prompting Baselines.** We compare against three single-call baselines: **Zero-Shot** without intermediate structure, **CoT** using natural-language reasoning (Wei et al., 2022; Kojima et al., 2022), and **PoT** using executable code (Chen et al., 2023b). We exclude multi-call methods from quantitative comparison because they require substantially more compute (3–100+ inference calls per problem); Table 5 instead compares design characteristics across both families. F-1's Phase 2 is a generic composable slot (Section 6); for these experiments we instantiate it with the same three solving strategies we use as baselines (Direct, CoT, and PoT), so that any gain over the baselines is attributable to Phase 1 rather than to a novel solver. This matched instantiation also makes the fixed-style variants (F-1+ZS, F-1+CoT, F-1+PoT) directly comparable to their corresponding baseline.

**Models.** We evaluate five reasoning models spanning proprietary diversity, open/proprietary frontier comparison, and within-family scale: GPT-5 and Gemini 2.5 Pro represent leading proprietary models from different vendors; DeepSeek-V3.1 provides an open frontier comparison; and Qwen3-235B and Qwen3-30B isolate scale within a shared reasoning-tuned family. This selection does not directly test smaller (<30B) or non-reasoning models, as discussed in Section 7.

# 5 Formula Prompting

Formula Prompting (FP) is a single-call prompt that asks the model to formalize a problem's governing equations before producing an answer, building on the pretraining-prevalence motivation of Section 3. We test the hypothesis that *explicit elicitation of governing equations improves LLM performance on applied-mathematics problems where domain-specific governing equations are identifiable*, and present four analyses that progressively isolate the contribution of the equation-formalization prefix: per-problem flip counts (Section 5.3), output-pattern composability (Section 5.4), variant accuracy at matched reasoning effort (Section 5.5), and failure-cause classification (Section 5.6).

## 5.1 Method

The FP prompt instructs the model to perform the following steps, in any order:

**Identify Givens.** Extract all explicitly stated values, conditions, and constraints. For example, in "A bank offers 5% annual interest compounded monthly," the givens include the rate (5%), the compounding frequency (monthly), and any principal or time mentioned.

**Identify Targets.** Determine what the problem asks to find or prove, clarifying the goal before attempting any solution strategy.

**Formulate Equations.** Express the mathematical relationships connecting givens to targets, committing to the problem's symbolic structure before committing to a solving strategy.

## 5.2 Prompt

**FP User Prompt**

```
{problem}
```

**Formalization:**
Identify givens and target. Write the key equations.

Solve the problem using the formalized equations.

Verify your solution, then put your final answer in \boxed{}.

The model receives this prompt once and produces the formalization, solution, and verification in a single response.

## 5.3 Per-Problem Contribution of Phase 1

We ask whether adding Phase 1 (equation formalization) to a fixed CoT solver flips more problems from incorrect to correct than the reverse. To isolate Phase 1, we compare CoT (the baseline) against F-1+CoT (the fixed-style variant of F-1 with Phase 2 constrained to CoT). The two differ only in the presence of the equation-formalization prefix. We analyze GPT-5 responses across all benchmarks (Table 2).

| Benchmark | F-1✓CoT× | CoT✓F-1× | Net | N |
|---|---|---|---|---|
| Olympiad (OE) | 48 | 39 | +9 | 910 |
| Olympiad (TP) | 70 | 50 | +20 | 528 |
| IMO Answer | 34 | 47 | −13 | 400 |
| IMO Proof | 15 | 7 | +8 | 60 |
| AICrypto† | 1 | 1 | 0 | 18 |
| **Total** | **168** | **144** | **+24** | **1,916** |

Table 2: Per-problem flip analysis (GPT-5): F-1✓CoT× = problems F-1+CoT solves but CoT fails; CoT✓F-1× = the reverse. Net positive indicates F-1+CoT's equation formalization flips more problems to correct. †n=18, interpret with caution.

**Equation formalization provides net positive contribution.** Across 1,916 problems, F-1+CoT flips 168 problems from incorrect to correct compared to CoT, while CoT flips only 144 in the reverse direction (net +24). The effect is strongest on OlympiadBench theorem-proving (net +20), where explicit equation writing helps structure multi-step proofs.

**Exception: IMO AnswerBench.** The net-positive pattern reverses on IMO AnswerBench, where F-1+CoT shows a net negative (−13). This reversal is consistent with our later finding that 90.7% of IMO problems are too difficult for all methods (Table 20); on the small fraction of problems where methods differ, the overhead of equation formalization occasionally introduces errors without compensating benefit, as competition mathematics problems often require creative insight rather than systematic equation manipulation.

**Phase 1 insight also generalizes to a PoT solver.** The same isolation logic applied to PoT (baseline) vs F-1+PoT (FP prefix + PoT solver) shows substantially larger flips: net +199 on OlympiadBench and +60 on IMO AnswerBench. That is, 199 OlympiadBench problems and 60 IMO AnswerBench problems that pure PoT solves incorrectly are correctly solved when an equation-formalization prefix is added before code generation, suggesting that the prefix unlocks problems pure code generation cannot reach.

**Paired statistical tests confirm the gain.** Pooling 9,163 paired per-problem outcomes across OlympiadBench and IMO-AnswerBench (the two benchmarks with binary per-problem correctness) over all 5 models, F-1+CoT significantly outperforms Zero-Shot (+7.39 pp), CoT (+1.58 pp), and PoT (+7.50 pp), all at $p<0.001$ (two-sided McNemar). Per-benchmark and per-model breakdowns and CIs appear in Section N.

### 5.4 Composability with Solving Strategies

A natural question is whether FP *replaces* a solving strategy with equation reasoning or *composes* as a structural prefix that leaves the downstream solver intact. We test this by running three FP-prefix variants on Qwen3-30B-Thinking-2507: FP followed by Zero-shot, by CoT, and by PoT solving. We label these **F-1+ZS**, **F-1+CoT**, and **F-1+PoT** for consistency with the fixed-style variant naming used later in Section 6. For each variant we extract per-output presence of **FP**, **CoT**, and **PoT markers** drawn from the same n-gram sets used in our pretraining analysis (Table 25 in Section L). We compute pattern presence on 1,456 problems spanning all four benchmarks. The goal is to determine whether the FP prefix survives composition while the solver's behavior is preserved.

**FP markers persist across all solving strategies; solver-specific markers cleanly isolate.** Equation-formalization markers appear in 43%, 32%, and 55% of F-1+ZS, F-1+CoT, and F-1+PoT outputs, showing that the FP prefix persists across solvers. These rates exceed baseline background levels: our output-validation analysis found FP markers at ≥10× higher frequency in FP than pure-CoT outputs (Section 3). PoT markers remain solver-specific, appearing in 69% of F-1+PoT outputs but 0% of F-1+ZS and F-1+CoT. This asymmetry (prefix retained, solver vocabulary isolated) suggests that FP acts as a structural prefix only rather than a replacement for the solver. "Step by step" appears in 54–61% of all variants, including F-1+PoT, supporting its status as generic English rather than CoT-specific. The full matrix appears in Figure 6 of Section M.

### 5.5 Variant Accuracy at Matched Reasoning Budget

Having shown that FP markers persist across solvers (Section 5.4), we next test whether this composability improves accuracy. We compare F-1+ZS, F-1+CoT, and F-1+PoT on three reasoning-tier LLMs: DeepSeek-V3.1 reasoner, GPT-5 high-reasoning, and Qwen3-30B-A3B-Thinking-2507. To prevent confounding solver choice with reasoning compute, we fix each model's reasoning budget at its maximum across variants. Phase 1 is identical across variants; Phase 2 differs by one instruction line.

Accuracy across three models, three variants, and four benchmarks is summarized in Table 16

(Section G); CryptoProof is the AICrypto subset reported as % of 330 total partial-credit points.

**F-1+ZS is consistently competitive; F-1+PoT underperforms on math-heavy benchmarks.** F-1+ZS is the strongest or tied-strongest variant on 9 of 12 (model, benchmark) cells; F-1+PoT is the weakest on math-heavy benchmarks for DeepSeek-V3.1 and GPT-5 (beaten by F-1+ZS by 10.4 pp on OB Math, DeepSeek, $p < 0.001$; and 7.2 pp on OB Physics, $p = 0.024$; full statistics in Section G). These results suggest that most gains come from the FP prefix itself: once given equation formalization, thinking-mode models often perform implicit step-by-step reasoning without additional CoT or PoT instructions. Forcing a code-based solve can disrupt this symbolic reasoning, consistent with Section 3. CryptoProof, which lacks a single governing equation, shows no consistent best variant across models, supporting the hypothesis that FP helps primarily when governing equations are identifiable.

### 5.6 Failure-Cause Classification

To diagnose *why* the F-1 prefix sometimes fails to convert into a correct answer, we sample 88 F-1 failures (stratified across (benchmark, model) cells) and have three judges from different model families (GPT-5.1, DeepSeek-V4-Pro thinking, and Claude Opus 4.7) classify each into one of three categories: (A) wrong Phase-1 equation → wrong answer; (B) wrong Phase-1 but recovered; (C) right Phase-1, downstream execution error. Multi-judge cross-validation defends against single-judge bias (Manakul et al., 2023). Inter-judge agreement is moderate, with pairwise Cohen's $\kappa$ ranging 0.31–0.52 and Fleiss' $\kappa$=0.403 ($n$=86; full table in Section O). All three judges agree on marginals: **C dominates at 73.3%** (majority-vote), A=25.6%, B=1.2%. **Phase 1 equation formalization is reliable; failures occur during downstream solving**: in nearly three-quarters of failure cases the governing equation is correct, validating the FP design even when the final answer is wrong. The 25.6% over-reliance is a known limitation (Section 7); the low self-correction rate (B=1.2%) is confirmed by a Phase-1 verification probe (Section P) that recovers only +0.50 pp on FinanceMath × GPT-5 and never explicitly revises a Phase-1 equation, pointing to multi-call backtracking (Yao et al., 2023; Liu et al., 2023) rather than single-call self-verification as the right mitigation.

## 6 Formula-One Prompting

Formula-One Prompting (F-1) extends FP (Section 5) by adding a composable second phase. In a single LLM generation, **Phase 1 reuses the FP formalization prompt**, and **Phase 2** selects among Direct calculation, CoT-style derivation, or PoT-style computation over the formalized equations. No external classifier or hand-written router is used.

### 6.1 Phase 2: Composable Solving

Phase 2 is a composable slot: once governing equations are formalized in Phase 1, any solving strategy can be invoked within the same generation. The prompt does not specify a fixed solving routine, nor does it encode explicit routing rules or decision criteria. Instead, the model selects a strategy in-context from the equations it produced, letting equation structure guide solving without over-constraining it. The exposed strategies and rationale appear in Section 4.

### 6.2 Prompt Design

F-1 uses a single prompt for all phases, generating intermediate reasoning and the final answer in one call. It extends the FP prompt (Section 5) by replacing direct solving with a composable-solving menu:

**System Prompt (Unified Core)**

```
You are an AI assistant that solves problems
mainly through equations.
```

**F-1 User Prompt Template**

```
{problem}
```

**Phase 1 (Formalization):**
Identify givens and target. Write the key equations.

**Phase 2 (Composable Solving):**
Solve via CoT (think step-by-step), PoT (compute with code), direct calculation, or elimination.

Verify your solution, then put your final answer in \boxed{}.

The model receives one prompt and returns formalization, strategy selection, and solution in a single response, with verification against the formalized equations. For composability analysis (Section 5.5), we also evaluate fixed-style variants (F-1+ZS, F-1+CoT, F-1+PoT) that share Phase 1 but constrain Phase 2; F-1+ZS is equivalent to FP. Full templates appear in Section B.

| Benchmark | Method | Proprietary | | Open Source | | | Overall |
|---|---|---|---|---|---|---|---|
| | | GPT-5 | Gemini 2.5 Pro | Qwen3-30B | Qwen3-235B | DeepSeek-V3.1 | |
| IMO-Bench | Zero-Shot | **56.26** | 55.44 | 50.99 | 21.85 | **30.54** | **43.02** |
| | CoT | 54.71 | 53.91 | **53.35** | 20.45 | 27.56 | 42.00 |
| | PoT | 47.73 | 49.06 | 42.61 | **23.94** | 17.48 | 36.16 |
| | **F-1 (Ours)** | 55.58 | **57.02** | 51.82 | 21.84 | 27.64 | 42.78 |
| OlympiadBench | Zero-Shot | 58.77 | 41.97 | 51.67 | 47.33 | 45.24 | 49.00 |
| | CoT | 63.12 | 57.67 | 58.23 | 48.03 | **52.42** | 55.89 |
| | PoT | 43.72 | 46.29 | 48.22 | 38.56 | 33.99 | 42.16 |
| | **F-1 (Ours)** | **65.81** | **59.70** | **59.68**$^{***}$ | **52.75**$^{***}$ | 50.29 | **57.65** |
| FinanceMath | Zero-Shot | 26.00 | 32.50 | 28.50 | 35.50 | 27.50 | 30.00 |
| | CoT | 42.50 | 56.00 | 53.50 | 35.00 | 28.00 | 43.00 |
| | PoT | 54.00 | 55.50 | 53.50 | 59.50 | 37.50 | 52.00 |
| | **F-1 (Ours)** | **64.00**$^{***}$ | **56.50** | **56.00** | **61.00**$^{***}$ | **44.00**$^{***}$ | **56.30** |
| AICrypto | Zero-Shot | 91.10 | 81.50 | 74.70 | 65.20 | 67.30 | 75.96 |
| | CoT | **98.60** | 93.00 | 65.80 | 71.50 | 72.60 | 80.30 |
| | PoT | 88.70 | 92.10 | 77.70 | 65.60 | 77.10 | 80.24 |
| | **F-1 (Ours)** | 98.50 | **96.70** | **85.80** | **76.50** | **80.20** | **87.54** |
| **Overall** | Zero-Shot | 58.03 | 52.85 | 51.47 | 42.47 | 42.65 | 49.49 |
| | CoT | 64.73 | 65.15 | 57.72 | 43.74 | 45.15 | 55.30 |
| | PoT | 58.54 | 60.74 | 55.51 | 46.90 | 41.52 | 52.64 |
| | **F-1 (Ours)** | **70.97** | **67.48** | **63.33** | **53.02** | **50.53** | **61.06** |

Table 3: Main results showing accuracy percentage across benchmarks and models. **Overall** is unweighted macro-averaged across the four benchmarks. Best results per benchmark-model are in **bold**. Markers on F-1 cells indicate F-1 vs CoT McNemar two-sided $p$-values: $^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$. Full per-combination breakdown for all three baseline comparisons is available in Table 27.

### 6.3 Main Results

**Main Results.** F-1 beats all three baselines in 14 of 20 model-benchmark cells and CoT specifically in 17 of 20 (Table 3). On macro-average across the four benchmarks, F-1 outperforms CoT by $+5.76$ pp and PoT by $+8.42$ pp. McNemar's two-sided test on paired per-problem predictions confirms the effect: F-1 vs CoT is significant ($p<0.05$) on 5 of 20 cells (3 of 5 FinanceMath, 2 of 5 OlympiadBench); F-1 vs Zero-Shot on 10 of 20 (all 5 FinanceMath, 4 of 5 OlympiadBench, 1 of 5 AICrypto); F-1 vs PoT on 11 of 20 (5 of 5 IMO-Bench, 5 of 5 OlympiadBench, 1 of 5 FinanceMath). AICrypto's per-cell $n=18$ limits per-cell power, but pooling all 5 models restores significance for F-1 vs CoT ($+6.45$ pp, $p=0.022$) and F-1 vs Zero-Shot ($+10.49$ pp, $p<0.001$).

**Gains concentrate on applied domains.** F-1 lifts CoT by $+13.30$ pp on FinanceMath and $+7.24$ pp on AICrypto, the two benchmarks most directly suited to governing-equation formulation. The within-OlympiadBench subtask breakdown below further supports this pattern.

**The effect does not require frontier scale.** The smallest model in our suite, Qwen3-30B, gains $+5.6$ pp over CoT, comparable to frontier models. F-1 also automatically shifts its Phase 2 choice by problem type, selecting PoT on computation-heavy benchmarks (FinanceMath) and CoT on reasoning-heavy ones (AICrypto). Detailed per-subtask breakdowns are in Section D.

**Head-to-head vs Formulate-and-Solve.** We compare F-1 against the closest prior single-call equation-first method, Formulate-and-Solve (Kao et al., 2024), on FinanceMath ($n=200$) and OlympiadBench Math ($n=674$) with GPT-5 and Qwen3-30B (Table 4). Aggregate accuracies are close ($\pm 5$ pp), but 7.6–15.7% of problems are solved by exactly one method – F-1's composable Phase 2 is qualitatively distinct from F&S's fixed algebraic solve, not a marginal improvement on the same distribution. Full cross-tab in Section Q.

| Benchmark | Model | F&S | F-1 | Disjoint |
|---|---|---|---|---|
| FinanceMath | GPT-5 | 64.00 | 64.00 | 11.0 |
| FinanceMath | Qwen3-30B | 51.50 | **56.00** | 14.5 |
| OB Math | GPT-5 | **87.09** | 86.35 | 7.6 |
| OB Math | Qwen3-30B | 78.64 | 78.64 | 15.7 |

Table 4: F-1 vs Formulate-and-Solve (Kao et al., 2024) accuracy (%). **Disjoint** = % problems solved by exactly one method.

### 6.4 Subtask Drill-Down: Math vs Physics

As a within-benchmark probe of the applied-math hypothesis (Section 5), we split OlympiadBench into competition mathematics and applied physics subtasks. Competition mathematics typically requires ad-hoc per-problem equation systems, whereas applied physics typically invokes canonical governing equations (Newton's laws, thermodynamic relations). Using GPT-5 as a single-model probe, F-1's gains over CoT are larger on applied physics ($+2.55$ pp open-ended, $+4.00$ pp theorem-proving) than on competition mathematics ($+0.44$ pp open-ended, $+3.77$ pp theorem-proving). The two subsets are not matched for difficulty, topic coverage, or problem style, so this is suggestive rather than controlled evidence, but consistent with the hypothesis. Full per-model results are in Table 12.

### 6.5 Token Efficiency

F-1 tops the accuracy-token efficiency frontier (Table 21). Using Efficiency Ratio = (Accuracy / Avg tokens) $\times$ 100, F-1 averages **1.51** across the four benchmarks, versus Zero-Shot 1.20, CoT 1.26, and PoT 1.24. The Phase-1 equation prefix adds only $\sim$68 prompt tokens over Zero-Shot, so the accuracy lift comes with little compute overhead. Per-model breakdowns by benchmark and a tokens-per-correct summary appear in Section C; overall win/tie/loss counts in Section J.

### 6.6 Strategy Distribution

F-1's Phase 2 selects a solving style per problem without an external classifier. Post-hoc classifying each F-1 response into Direct/CoT/PoT/Hybrid (Table 18 in Section H) reveals clear domain adaptation: FinanceMath elicits the highest PoT usage (31.4%) due to numerical computation, AICrypto uses CoT and Hybrid exclusively (0% PoT) for proof reasoning, and IMO-Bench shows the highest Hybrid usage (65.3%).

A complementary per-problem outcome analysis (Table 20 in Section J) defines *selection accuracy* as F-1's success rate on problems where strategy choice matters: FinanceMath 73.0%, OlympiadBench 69.9%, IMO-Bench 61.0%, consistent with the applied-math hypothesis. The "F-1 Only" subset (only F-1 succeeds, all baselines fail) reaches 5.5% on FinanceMath and 7.8% on AICrypto; F-1 attains 80.9–84.1% of the single-call ceiling (Table 22). Across 60 model-benchmark comparisons, F-1 wins 53/60 (88.3%; Table 23). On IMO-Bench, 90.7% of problems are too difficult for all methods (Table 20), leaving little headroom for strategy selection; F-1 still beats PoT by $+6.62$ pp. Three qualitative cases (Section E) illustrate F-1 mechanisms: preventing semantic confusion in physics (Lorentz transformations), enforcing format consistency in finance (decimal output), and structured proof construction in cryptography (PRF security game).

### 6.7 Domain Boundary: Negative Control on a Non-Equation Task

To test F-1 outside its intended domain (Section 5), we apply it to BBH word_sorting (Suzgun et al., 2023) ($n$=250), which lacks quantitative or equation structure. F-1 regresses substantially relative to CoT: $-49.2$ pp on GPT-5 (46.8% vs 96.0%), $-42.4$ pp on Gemini-2.5-Pro (56.0% vs 98.4%), and $-4.8$ pp on Qwen3-30B-Thinking (88.4% vs 93.2%). Instruction-following models often interpret the formalization directive literally, attempting to express sorting as equations, while the thinking-mode model adapts better. These results support our hypothesis that F-1 helps when governing equations are identifiable, but can impose harmful overhead otherwise. Thus, F-1 is best viewed as a targeted applied-math tool rather than a general-purpose prompting upgrade. Details and failures appear in Section R.

## 7 Conclusion

We presented Formula Prompting (FP) and Formula-One Prompting (F-1), single-call methods that elicit governing equations before solving applied-math problems. FP improves per-problem outcomes over CoT, persists across solvers, and is competitive with the fixed-style variants; 73.3% of its remaining failures occur downstream of a correct Phase-1 equation. F-1 extends FP with a composable Phase 2 menu, outperforming CoT by 5.76 pp and PoT by 8.42 pp on average (with the largest gain of 13.30 pp on FinanceMath), while attaining the highest token-efficiency ratio (1.51 vs 1.20–1.26). Gains concentrate in applied-math domains with identifiable governing equations and do not transfer to proof-oriented or non-equation tasks. A cleaner isolation of the equation-formalization effect on non-equation domains (FP only), and a mechanistic account of why equation elicitation changes model behavior, remain open.

## Limitations

**Over-reliance on Phase-1 equations.** The failure-cause analysis (§5.6) shows that 25.6% of F-1 failures are wrong-equation → wrong-answer (majority-vote across three independent judges, $n$=86). Models commit to whatever Phase 1 produces – self-correction is rare (1.2%). We tested whether an explicit Phase-1 self-verification step within the same call mitigates this (Section P): on FinanceMath × GPT-5 ($n$=200), F-1+Verify yields only +0.50 pp (net +1, McNemar $p$=1.0) and never explicitly revises a Phase-1 equation, despite a +28.8% token overhead. Practitioners should treat F-1 outputs as conditional on Phase 1 quality; multi-call backtracking (Yao et al., 2023; Liu et al., 2023), not single-call self-verification, is the more promising mitigation.

**Domain boundary – F-1 should not be deployed on non-equation tasks.** The BBH negative control (§6.7) shows that F-1 *regresses* sharply (−42 to −49 pp on GPT-5 and Gemini-2.5-Pro) when applied to a sorting task with no equation structure. The Phase-1 directive forces models to formalize a non-formalizable problem, derailing reasoning. F-1 is designed for and validated on applied-math domains; it must not be applied as a generic prompting upgrade.

**Sample size and model scope.** Our evaluation spans 30B to frontier scale and does not include smaller (7B/13B) or non-reasoning models. Some benchmarks are small due to data scarcity (AICrypto n=18, OlympiadBench TP_physics n=25); main conclusions are supported by larger subsets (Lu et al., 2026; Yang et al., 2024). AICrypto's small size limits per-model statistical power (minimum detectable effect 19.14 pp at the median model); we mitigate by reporting partial-credit per-problem scores and pooling 90 paired observations across 5 models, which restores significance for F-1 vs all three baselines (Table 28). The CryptoProof regression (§5.5) further illustrates the domain boundary: theorem proving lacks a single governing equation, and the same infini-gram pattern that tracks F-1's strength on applied math tracks its weakness here.

**Mechanism scope and extensions.** Our infini-gram analysis is correlational: it measures pretraining prevalence of method-associated patterns, not the latent circuits that determine whether the prefix actually exploits those patterns. We use it to motivate the design and bound the applicable domain, not as causal evidence. Mechanistic interpretability of which circuits respond to equation elicitation, and multi-call backtracking (Yao et al., 2023; Liu et al., 2023) extensions of the current single-call design, are left for future work.

## Ethics Statement

This work introduces a prompting technique for mathematical reasoning in LLMs. We do not foresee direct negative societal impacts, though enhancing mathematical reasoning capabilities may increase the risk of misuse such as automated cheating in educational contexts. We encourage responsible deployment with appropriate safeguards. More broadly, novel prompting techniques can elicit unexpected behaviors from LLMs; practitioners should remain aware of this inherent uncertainty when deploying such methods.

## References


Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023a. Frugalgpt: How to use large language models while reducing cost and improving performance. *Preprint*, arXiv:2305.05176.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. In *NeurIPS 2023 Workshop on Mathematical Reasoning*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *Preprint*, arXiv:2303.05398.

Kuei-Chun Kao, Ruochen Wang, and Cho-Jui Hsieh. 2024. Solving for X and beyond: Can large language models solve complex math problems with more-than-two unknowns? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16821–16843, Miami, Florida, USA. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Ayeong Lee, Ethan Che, and Tianyi Peng. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. *Preprint*, arXiv:2503.01141.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Preprint*, arXiv:2206.14858.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2025. Datacomp-lm: In search of the next generation of training sets for language models. *Preprint*, arXiv:2406.11794.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse X-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2822, Singapore. Association for Computational Linguistics.

Pan Lu, Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, and James Zou. 2026. Solving inequality proofs with large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. 2025. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35418–35442, Suzhou, China. Association for Computational Linguistics.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D. Hardy, and Thomas L. Griffiths. 2024. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41):e2322420121.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *Preprint*, arXiv:2310.06786.

Ben Prystawski, Michael Y. Li, and Noah D. Goodman. 2023. Why think step by step? reasoning emerges from the locality of experience. In *Advances in Neural Information Processing Systems*.

Yasaman Razeghi, Robert L. Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *Preprint*, arXiv:2409.12183.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2024. Scibench: Evaluating college-level scientific

problem-solving abilities of large language models. *Preprint*, arXiv:2307.10635.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yu Wang, Yijian Liu, Liheng Ji, Han Luo, Wenjie Li, Xiaofei Zhou, Chiyun Feng, Puji Wang, Yuhan Cao, Geyuan Zhang, Xiaojian Li, Rongwu Xu, Yilei Chen, and Tianxing He. 2025. Aicrypto: A comprehensive benchmark for evaluating cryptography capabilities of large language models. *Preprint*, arXiv:2507.09580.

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

Hao Xu, Jiacheng Liu, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2025a. Infini-gram mini: Exact n-gram search at the Internet scale with FM-index. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24944–24969, Suzhou, China. Association for Computational Linguistics.

Xin Xu, Qiyun Xu, Tong Xiao, Tianhao Chen, Yuchen Yan, Jiaxin Zhang, Shizhe Diao, Can Yang, and Yang Wang. 2025b. UGPhysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. In *Forty-second International Conference on Machine Learning*.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. 2024. Formal mathematical reasoning: A new frontier in ai. *Preprint*, arXiv:2412.16075.

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

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2024. Financemath: Knowledge-intensive math reasoning in finance domains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12841–12858, Bangkok, Thailand. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jianpeng Zhou, Wanjun Zhong, Yanlin Wang, and Jiahai Wang. 2024. Adaptive-solver framework for dynamic strategy selection in large language model reasoning. *Preprint*, arXiv:2310.01446.

# A Method Comparison

| Method | Calls | Fo? | Comp? | Repr. | Strategy | Cost |
|---|---|---|---|---|---|---|
| Zero-shot | 1 | ✗ | ✗ | – | Fixed | V.Low |
| CoT | 1 | ✗ | ✗ | NL | Fixed | Low |
| PoT | 1 | ✗ | ✗ | Code | Fixed | Low |
| Plan-Solve | 1 | ✗ | ✗ | Plan | Fixed | Low |
| Form-Solve | 1 | ✓ | ✗ | Eq. | Fixed | Low |
| ToT | 10-100+ | ✗ | ✗ | Tree | Search | V.High |
| GoT | 20-80 | ✗ | ✗ | Graph | Search | V.High |
| XoT | 3-10 | ✗ | ✗ | Mixed | Classifier | Med. |
| **F-1 (Ours)** | **1** | ✓ | ✓ | **Eq.** | **Composable** | **Low** |

Table 5: Comparison of prompting methods referenced in §2. "**Fo?**": explicit formalization? "**Comp?**": composable prefix that combines with any Phase 2 strategy (the same Phase 1 attaches to ZS/CoT/PoT). "**Repr.**": intermediate representation (NL: natural language, Eq.: equations). "**Cost**": computational cost. "✓": supported, "✗": not supported. Form-Solve = Formulate-and-Solve.

# B Prompt Templates

This appendix presents the complete prompt templates used for each benchmark. For all strategies, `{problem}` denotes the problem statement placeholder.

## B.1 IMO-Bench Prompts

### B.1.1 System Prompts

**Zero-Shot System**

```
(no system prompt)
```

**CoT System**

```
You are an AI assistant that solves problems
by thinking step-by-step.
```

**PoT System**

```
You are an AI assistant that solves problems
by generating Python code.
```

**F-1 System (Ours)**

```
You are an AI assistant that solves problems
mainly through equations.
```

### B.1.2 User Prompts

**Zero-Shot User**

```
{problem}

Put your final answer in \boxed{}.
```

**CoT User**

```
{problem}

Let's think step-by-step. Put your final answer in \boxed{}.
```

**PoT User**

```
{problem}

Write Python code to solve this. Use sympy/numpy/math as needed. Store the final answer in a variable called ‘answer‘ and print it. Return only the code, then put your final answer in \boxed{}.
```

**F-1 User (Ours)**

```
{problem}

Identify givens and target. Write the key equations. -> Solve via CoT (Think step-by-step), PoT (Compute with code), direct calculation, or elimination. -> Verify your solution, then put your final answer in \boxed{}.
```

### B.1.3 F-1 Variants (Composability)

The variants share the same Phase 1 (formalization) but fix Phase 2 (solving style). Used for the composability analysis (Section 5.5, Section G).

**F-1+ZS System**

```
You are an AI assistant that first formalizes governing equations before solving.
```

**F-1+ZS User**

```
{problem}

First, identify the key variables and write the governing equations in LaTeX. Then solve directly using those equations. Put your final answer in \boxed{}.
```

**F-1+CoT System**

```
You are an AI assistant that first formalizes governing equations before solving step-by-step.
```

**F-1+CoT User**

```
{problem}

First, identify the key variables and write the governing equations in LaTeX. Then let's think step-by-step to solve using those equations. Put your final answer in \boxed{}.
```

**F-1+PoT System**

```
You are an AI assistant that first formalizes governing equations before solving by writing Python code.
```

**F-1+PoT User**

```
{problem}

First, identify the key variables and write the governing equations in LaTeX. Then write Python code to implement those equations and solve. Use sympy/numpy/math as needed. Store the final answer in ‘answer‘ and print it. Return only the code, then put your final answer in \boxed{}.
```

## B.2 OlympiadBench Prompts

### B.2.1 System Prompts

**Zero-Shot / CoT / PoT System**

```
You are an AI assistant. Please answer the following Math competition problems as required.
```

**F-1 System (Ours)**

```
You are an AI assistant that solves problems mainly through equations. Please answer the following Math competition problems as required.
```

### B.2.2 User Prompts

**Zero-Shot User**

```
{problem_statement}

{answer_type_text}Please calculate the answer according to the given requirements and the information provided. Please end your solution with "So the final answer is {boxed_format}." and give the result explicitly{unit_text}.
```

**CoT User**

```
{problem_statement}

{answer_type_text}Please solve this problem step-by-step:
1. First, carefully read and understand the problem
2. Identify what is given and what needs to be found
3. Break down the problem into smaller steps
4. Solve each step systematically, showing your reasoning
5. Combine the results of each step to arrive at the final answer

Please end your solution with "So the final answer is {boxed_format}."
```

**PoT User**

```
{problem_statement}

{answer_type_text}Please solve this problem by:
1. Writing Python code to solve the problem programmatically
2. Showing your code and explaining each step
3. Running the code to get the numerical answer
```

You can use libraries like math, numpy, sympy, etc.

**F-1 User (Ours)**

{problem_statement}

{answer_type_text}
Please solve using an Equation-First approach:

1. List key variables and state the target quantity.
2. Write the main governing equations/identities in LaTeX first.
3. After the equations, choose any suitable solving style:
- CoT: think step-by-step, solve systematically, then combine results.
- PoT: write Python code to implement the equations.
- Zero-Shot: derive directly from equations with minimal text.
4. Combine the results of each step to arrive at the final answer

Please end your solution with "So the final answer is {boxed_format}."

### B.2.3 F-1 Variants (Composability)

The variants share the same Phase 1 (formalization) but fix Phase 2 (solving style). Used for the composability analysis (Section 5.5, Section G).

**F-1+ZS System**

You are an AI assistant solving math/physics competition problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then solve by direct substitution. Please answer the following problems as required.

**F-1+ZS User**

{problem_statement}

{answer_type_text}Phase 1 - Formalize:
1. List the key variables (givens with their values, and the unknown to find).
2. Write the governing equation(s) in LaTeX using \begin{equation}...\end{equation}.

Phase 2 - Solve: Substitute the values into the equations and compute the final answer directly.

Please end your solution with "So the final answer is {boxed_format}." and give the result explicitly{unit_text}.

**F-1+CoT System**

You are an AI assistant solving math/physics competition problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then solve through step-by-step reasoning. Please answer the following problems as required.

**F-1+CoT User**

{problem_statement}

{answer_type_text}Phase 1 - Formalize: (same as F-1+ZS).

Phase 2 - Solve: Using the equations above, work through the problem step-by-step, showing each manipulation that leads to the final answer.

Please end your solution with "So the final answer is {boxed_format}." and give the result explicitly{unit_text}.

**F-1+PoT System**

You are an AI assistant solving math/physics competition problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then implement those equations as a Python program. Please answer the following problems as required.

**F-1+PoT User**

{problem_statement}

{answer_type_text}Phase 1 - Formalize: (same as F-1+ZS).

Phase 2 - Solve: Implement the equations above as Python code (sympy/numpy/math). Store the final answer in ‘answer‘ and print it.

Please end your solution with "So the final answer is {boxed_format}."

## B.3 FinanceMath Prompts

### B.3.1 System Prompts

**Zero-Shot System**

You are a financial expert, you are supposed to answer the given question. Therefore, the answer is {final answer}. The final answer should be a numeric value (3 decimal).

**CoT System**

You are a financial expert, you are supposed to answer the given question. You need to first think through the problem step-by-step, documenting each necessary step. Then you are required to conclude your response with the final answer in your last sentence as ’Therefore, the answer is {final answer}’. The final answer should be a numeric value (3 decimal).

**PoT System**

You are a financial expert, you are supposed to generate a Python program to answer the given question. The returned value of the program is supposed to be the answer. Here is an example of the Python program:

“‘python
def solution():

```
# Define variables name and value
revenue = 600000
avg_account_receivable = 50000
# Do math calculation to get the answer
receivables_turnover = revenue / avg_account_receivable
answer = 365 / receivables_turnover
# return answer
return answer
“‘
Then you are required to conclude your response with the final answer in your last sentence as Therefore, the answer is {final answer}’. The final answer should be a numeric value (3 decimal).
```

**F-1 System (Ours)**

```
You are a financial expert. Solve problems mainly through equations.
OUTPUT CONTRACT (strict): The very last line MUST be exactly:
Final Answer (3 decimal) : <number>
```

### B.3.2 User Prefix (F-1 only)

**F-1 User Prefix**

```
Write equations (LaTeX) with minimal text; show steps clearly.
```

### B.3.3 F-1 Variants (Composability)

**F-1+ZS System**

```
You are a financial expert. Solve problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then solve by direct substitution.
OUTPUT CONTRACT (strict): The very last line MUST be exactly: ’Therefore, the answer is <number>’, where <number> is a numeric value rounded to 3 decimals.
```

**F-1+ZS User Prefix**

```
Phase 1 - Formalize:
1. List the key variables (givens with their values, and the unknown to find).
2. Write the governing equation(s) in LaTeX using \begin{equation}...\end{equation}.
Phase 2 - Solve: Substitute the values into the equations and compute the final answer directly.
```

**F-1+CoT System**

```
You are a financial expert. Solve problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then solve through step-by-step reasoning. (Same OUTPUT CONTRACT as F-1+ZS.)
```

**F-1+CoT User Prefix**

```
Phase 1 - Formalize: (same as F-1+ZS).
Phase 2 - Solve: Using the equations above, work through the problem step-by-step, showing each algebraic manipulation.
```

**F-1+PoT System**

```
You are a financial expert. Solve problems using an equation-first approach: first formalize the problem with explicit key variables and governing equations, then implement those equations as a Python program. The Python program must define a function ‘solution()‘ that returns the numeric answer. After the code, conclude with the final answer. (Same OUTPUT CONTRACT as F-1+ZS.)
```

**F-1+PoT User Prefix**

```
Phase 1 - Formalize: (same as F-1+ZS).
Phase 2 - Solve: Using the equations above, write a Python ‘solution()‘ function that computes the answer.
```

## B.4 AICrypto Prompts

### B.4.1 Base System Prompt (All Strategies)

**Base System (All Methods)**

```
You are an expert cryptographer tasked with solving cryptographic proof problems. Your responses must demonstrate deep understanding of cryptographic principles, mathematical rigor, and clear logical reasoning.

Output Format Requirements:
Your response MUST be structured into exactly two sections:

## Analysis
- Present your complete thought process
- Show all intermediate steps and considerations
- This section is for your working/scratch work

## Proof
- Provide a clean, formal proof suitable for academic submission
- This section alone will be graded
```

### B.4.2 Strategy-Specific System Additions

**Zero-Shot Addition**

```
(no addition)
```

**CoT Addition**

```
Approach: Solve problems by thinking step-by-step.
```

**PoT Addition**

```
Approach: Use algorithmic thinking and construct explicit adversaries/reductions with pseudocode.
```

**F-1 Addition (Ours)**

```
Approach: Solve problems mainly through equations and formal mathematical derivations.
```

### B.4.3 User Message Prefixes

**Zero-Shot Prefix**

```
(none)
```

**CoT Prefix**

```
Let's think step-by-step.
```

**PoT Prefix**

```
Construct an explicit algorithm/adversary to solve this.
```

**F-1 Prefix (Ours)**

```
Solve using equations: identify givens -> write equations -> solve -> verify.
```

### B.4.4 F-1 Variants (Composability)

The variants append different system additions and user prefixes onto the shared base system prompt.

**F-1+ZS System Addition**

```
Approach: Use an equation-first method. First formalize the problem with explicit key variables (parameters, oracles, security goals) and governing equations/security definitions, then solve by direct application of the formalization.
```

**F-1+ZS User Prefix**

```
Phase 1 - Formalize:
1. List the key variables (parameters, oracles, security goals).
2. Write the governing equation(s)/security definition(s) using \begin{equation}...\end{equation}.
Phase 2 - Solve: Apply the formalization directly to derive the result.
```

**F-1+CoT System Addition**

```
Approach: Use an equation-first method. First formalize the problem with explicit key variables (parameters, oracles, security goals) and governing equations/security definitions, then derive the result through step-by-step reasoning.
```

**F-1+CoT User Prefix**

```
Phase 1 - Formalize: (same as F-1+ZS).
Phase 2 - Solve: Using the formalization above, derive the result through step-by-step reasoning.
```

**F-1+PoT System Addition**

```
Approach: Use an equation-first method. First formalize the problem with explicit key variables (parameters, oracles, security goals) and governing equations/security definitions, then construct an explicit algorithm/adversary/reduction in pseudocode.
```

**F-1+PoT User Prefix**

```
Phase 1 - Formalize: (same as F-1+ZS).
Phase 2 - Solve: Using the formalization above, construct an explicit algorithm/adversary/reduction in pseudocode.
```

## C Token Efficiency Analysis

This appendix provides detailed token efficiency analysis across models and benchmarks. We use two complementary metrics following Chen et al. (2023a) and Lee et al. (2025):

### Metrics Definition

- **Efficiency Ratio** = $\frac{\text{Accuracy}}{\text{Avg Tokens}} \times 100$ (measures accuracy points gained per 100 tokens; higher is better)
- **Tokens per Correct** = $\frac{\text{Avg Tokens}}{\text{Accuracy}/100}$ (measures average tokens needed per correct answer; lower is better)

These metrics capture the accuracy-efficiency tradeoff central to practical LLM deployment, where token count determines both latency and operational cost.

### C.1 FinanceMath Token Efficiency by Model

Table 6 shows token efficiency for FinanceMath (n=200). F-1 achieves the best efficiency ratio for 4 out of 5 models.

| | Avg Tokens | | | | Efficiency Ratio ↑ | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | ZS | CoT | PoT | F-1 | ZS | CoT | PoT | F-1 |
| GPT-5 | 2,000 | 2,239 | 2,668 | 2,271 | 1.30 | 1.90 | 2.02 | **2.82** |
| Gemini-2.5-Pro | 2,279 | 2,850 | 2,831 | 2,879 | 1.43 | 1.96 | 1.89 | **1.95** |
| Qwen3-30B | 2,708 | 3,987 | 3,502 | 3,478 | 1.05 | 1.34 | 1.53 | **1.61** |
| Qwen3-235B | 2,394 | 2,654 | 3,214 | 3,364 | 1.48 | 1.32 | **1.85** | 1.81 |
| DeepSeek-V3.1 | 598 | 571 | 711 | 627 | 4.60 | 4.90 | 5.28 | **7.02** |
| **Average** | 1,996 | 2,460 | 2,585 | 2,524 | 1.97 | 2.29 | 2.51 | **3.04** |

Table 6: FinanceMath (n=200) token efficiency by model. ZS=Zero-Shot.

### C.2 OlympiadBench Token Efficiency by Model

Table 7 shows token efficiency for OlympiadBench (n=1,438), averaged across all subjects.

| Model | Avg Tokens | | | | Efficiency Ratio ↑ | | | |
|---|---|---|---|---|---|---|---|---|
| | ZS | CoT | PoT | F-1 | ZS | CoT | PoT | F-1 |
| GPT-5 | 6,742 | 8,593 | 5,947 | 8,204 | 1.27 | 1.12 | 0.96 | **1.23** |
| Gemini-2.5-Pro | 17,645 | 17,144 | 15,736 | 15,973 | 0.37 | 0.64 | 0.47 | **0.72** |
| Qwen3-30B | 13,980 | 12,130 | 9,099 | 10,675 | 0.40 | 0.57 | 0.53 | **0.73** |
| Qwen3-235B | 16,849 | 17,103 | 12,434 | 12,970 | 0.31 | 0.38 | 0.48 | **0.65** |
| DeepSeek-V3.1 | 2,624 | 2,617 | 2,965 | 2,376 | 2.14 | 1.73 | 1.10 | **3.71** |
| **Average** | 11,221 | 11,517 | 9,236 | 10,040 | 0.97 | 0.99 | 0.70 | **1.08** |

Table 7: OlympiadBench (n=1,438) token efficiency by model.

### C.3 IMO-Bench Token Efficiency by Model

Table 8 shows token efficiency for IMO-Bench (n=460). Note that PoT shows slightly higher efficiency due to shorter outputs, though accuracy remains lower than F-1.

| Model | Avg Tokens | | | | Efficiency Ratio ↑ | | | |
|---|---|---|---|---|---|---|---|---|
| | ZS | CoT | PoT | F-1 | ZS | CoT | PoT | F-1 |
| GPT-5 | 17,578 | 17,787 | 14,665 | 17,159 | 0.27 | 0.26 | **0.30** | 0.27 |
| Gemini-2.5-Pro | 8,885 | 10,023 | 9,495 | 10,308 | 1.09 | 1.09 | **1.32** | 1.18 |
| Qwen3-30B | 20,986 | 20,112 | 16,012 | 19,794 | 0.19 | 0.21 | 0.21 | **0.21** |
| Qwen3-235B | 16,119 | 16,124 | 14,931 | 16,086 | 0.14 | 0.13 | **0.16** | 0.14 |
| DeepSeek-V3.1 | 3,444 | 3,409 | 2,632 | 3,594 | **0.93** | 0.85 | 0.74 | 0.82 |
| **Average** | 14,670 | 14,715 | 12,421 | 14,540 | 0.48 | 0.47 | **0.52** | 0.49 |

Table 8: IMO-Bench (n=460) token efficiency by model.

### C.4 AICrypto Token Efficiency by Model

Table 9 shows token efficiency for AICrypto (n=18). GPT-5 data is unavailable for this benchmark.

| Model | Avg Tokens | | | | Efficiency Ratio ↑ | | | |
|---|---|---|---|---|---|---|---|---|
| | ZS | CoT | PoT | F-1 | ZS | CoT | PoT | F-1 |
| Gemini-2.5-Pro | 3,367 | 5,105 | 5,000 | 5,288 | 2.42 | 1.82 | 1.84 | **1.83** |
| Qwen3-30B | 9,768 | 9,466 | 12,092 | 9,800 | 0.76 | 0.70 | 0.64 | **0.88** |
| Qwen3-235B | 9,909 | 8,570 | 8,360 | 9,282 | 0.66 | **0.83** | 0.78 | 0.82 |
| DeepSeek-V3.1 | 4,027 | 3,872 | 4,789 | 3,754 | 1.67 | 1.87 | 1.61 | **2.14** |
| **Average** | 6,768 | 6,753 | 7,560 | 7,031 | 1.38 | 1.31 | 1.22 | **1.42** |

Table 9: AICrypto (n=18) token efficiency by model. GPT-5 data unavailable.

### C.5 Tokens per Correct Answer Summary

Table 10 summarizes tokens per correct answer across all benchmarks. F-1 requires 23.2% fewer tokens than Zero-Shot and 11.7% fewer than CoT on average.

| Method | FinMath | Olympiad | IMO | Crypto | Avg |
|---|---|---|---|---|---|
| Zero-Shot | 6,624 | 51,929 | 51,192 | 9,597 | 29,836 |
| CoT | 5,486 | 37,649 | 51,355 | 9,299 | 25,947 |
| PoT | 4,815 | 37,120 | 43,638 | 9,986 | 23,890 |
| **F-1 (Ours)** | **4,368** | **25,517** | 53,376 | **8,426** | **22,922** |

Table 10: Tokens per Correct answer (lower is better).

## D Additional Results by Subtask

This appendix provides detailed results broken down by subtasks within each benchmark. Overall scores are reported in Table 3.

| Model | ZS | CoT | PoT | F-1 |
|---|---|---|---|---|
| *IMO-AnswerBench* | | | | |
| GPT-5 | 67.75 | **67.75** | 49.50 | 64.50 |
| Gemini 2.5 Pro | 68.50 | 69.25 | 60.50 | **69.75** |
| Qwen3-30B | **61.50** | 60.75 | 54.50 | 61.25 |
| Qwen3-235B | 7.75 | 8.75 | **15.50** | 7.25 |
| DeepSeek-V3.1 | **42.75** | 42.50 | 24.00 | 41.00 |
| *IMO-ProofBench* | | | | |
| GPT-5 | 44.76 | 41.67 | 45.95 | **46.67** |
| Gemini 2.5 Pro | 42.38 | 38.57 | 37.62 | **44.29** |
| Qwen3-30B | 40.48 | **45.95** | 30.71 | 42.38 |
| Qwen3-235B | 35.95 | 32.14 | 32.38 | **36.43** |
| DeepSeek-V3.1 | **18.33** | 12.62 | 10.95 | 14.29 |

Table 11: IMO-Bench breakdown by subtask (%). ZS = Zero-Shot. Best per row in **bold**.

| Model | ZS | CoT | PoT | F-1 |
|---|---|---|---|---|
| *OE_maths (n=674)* | | | | |
| GPT-5 | **87.24** | 85.91 | 77.89 | 86.35 |
| Gemini 2.5 Pro | 31.90 | 83.23 | 86.05 | **86.20** |
| Qwen3-30B | 61.42 | 70.92 | 75.67 | **78.64** |
| Qwen3-235B | 70.40 | 76.05 | 70.83 | **77.61** |
| DeepSeek-V3.1 | **75.07** | 74.63 | 70.18 | 72.11 |
| *OE_physics (n=236)* | | | | |
| GPT-5 | **46.61** | 42.37 | 40.68 | 44.92 |
| Gemini 2.5 Pro | 55.93 | 53.81 | 50.00 | **56.36** |
| Qwen3-30B | 40.68 | **43.22** | 39.41 | 38.14 |
| Qwen3-235B | **36.60** | 36.32 | 29.66 | **36.60** |
| DeepSeek-V3.1 | 41.95 | **48.73** | 39.83 | 44.92 |
| *TP_maths (n=503)* | | | | |
| GPT-5 | 29.22 | 32.21 | 12.33 | **35.98** |
| Gemini 2.5 Pro | 24.06 | **25.65** | 13.12 | 24.25 |
| Qwen3-30B | 36.58 | 38.77 | 33.80 | **41.95** |
| Qwen3-235B | 6.31 | 11.74 | 17.76 | **24.80** |
| DeepSeek-V3.1 | 7.95 | 10.34 | 5.96 | **12.13** |
| *TP_physics (n=25)* | | | | |
| GPT-5 | 72.00 | 92.00 | 44.00 | **96.00** |
| Gemini 2.5 Pro | 56.00 | 68.00 | 36.00 | **72.00** |
| Qwen3-30B | 68.00 | **80.00** | 44.00 | **80.00** |
| Qwen3-235B | **76.00** | 68.00 | 36.00 | 72.00 |
| DeepSeek-V3.1 | 56.00 | **76.00** | 20.00 | 72.00 |

Table 12: OlympiadBench breakdown by subtask (%). OE = Open-Ended, TP = Theorem-Proving. Best per row in **bold**.

### D.1 IMO-Bench

Table 11 shows results on IMO-Bench subtasks: IMO-AnswerBench (closed-form answers) and IMO-ProofBench (proof-based).

### D.2 OlympiadBench

Table 12 shows results on OlympiadBench subtasks: OE (open-ended) and TP (theorem-proving) for both mathematics and physics.

| Model | ZS | CoT | PoT | F-1 |
|---|---|---|---|---|
| *Accounting (n=39)* | | | | |
| GPT-5 | 25.64 | 58.97 | 61.54 | **76.92** |
| Gemini 2.5 Pro | 25.64 | 58.97 | 61.54 | **76.92** |
| Qwen3-30B | 35.90 | 61.54 | 64.10 | **66.67** |
| Qwen3-235B | 51.28 | 41.03 | 66.67 | **74.36** |
| DeepSeek-V3.1 | 20.51 | 23.08 | **58.97** | 43.59 |
| *Asset Classes & Derivatives (n=63)* | | | | |
| GPT-5 | 26.98 | 33.33 | 46.03 | **63.49** |
| Gemini 2.5 Pro | 28.57 | **52.38** | 46.03 | **52.38** |
| Qwen3-30B | 19.05 | **49.21** | **49.21** | **49.21** |
| Qwen3-235B | 22.22 | 26.98 | **60.32** | 57.14 |
| DeepSeek-V3.1 | 22.22 | 22.22 | 25.40 | **41.27** |
| *Corporate & Securities (n=10)* | | | | |
| GPT-5 | 20.00 | 40.00 | 60.00 | **70.00** |
| Gemini 2.5 Pro | 30.00 | **60.00** | 50.00 | **60.00** |
| Qwen3-30B | 20.00 | **60.00** | **60.00** | **60.00** |
| Qwen3-235B | 30.00 | 20.00 | **60.00** | **60.00** |
| DeepSeek-V3.1 | 10.00 | 20.00 | 20.00 | **70.00** |
| *Market Analysis & Economic (n=24)* | | | | |
| GPT-5 | 37.50 | 50.00 | 62.50 | **70.83** |
| Gemini 2.5 Pro | 45.83 | 54.17 | **66.67** | 54.17 |
| Qwen3-30B | 41.67 | 54.17 | 54.17 | **62.50** |
| Qwen3-235B | 54.17 | 45.83 | **62.50** | **62.50** |
| DeepSeek-V3.1 | 41.67 | 37.50 | **58.33** | 41.67 |
| *Portfolio Management (n=19)* | | | | |
| GPT-5 | 15.79 | 42.11 | **57.89** | **57.89** |
| Gemini 2.5 Pro | 36.84 | **63.16** | 57.89 | **63.16** |
| Qwen3-30B | 21.05 | 47.37 | **57.89** | **57.89** |
| Qwen3-235B | 36.84 | 36.84 | 47.37 | **68.42** |
| DeepSeek-V3.1 | 31.58 | 26.32 | **42.11** | 36.84 |
| *Quantitative Analysis (n=36)* | | | | |
| GPT-5 | 25.00 | 41.67 | 58.33 | **61.11** |
| Gemini 2.5 Pro | 38.89 | 55.56 | **58.33** | 55.56 |
| Qwen3-30B | 36.11 | **61.11** | 55.56 | 58.33 |
| Qwen3-235B | 33.33 | 41.67 | **63.89** | 61.11 |
| DeepSeek-V3.1 | 38.89 | 41.67 | 30.56 | **52.78** |
| *Risk Management (n=9)* | | | | |
| GPT-5 | **22.22** | **22.22** | **22.22** | 11.11 |
| Gemini 2.5 Pro | **22.22** | 11.11 | 11.11 | 11.11 |
| Qwen3-30B | **22.22** | **22.22** | 11.11 | **22.22** |
| Qwen3-235B | **22.22** | **22.22** | **22.22** | 11.11 |
| DeepSeek-V3.1 | **22.22** | **22.22** | 11.11 | **22.22** |

Table 13: FinanceMath breakdown by category (%). Best per row in **bold**.

### D.3 FinanceMath

Table 13 shows results across FinanceMath categories.

### D.4 AICrypto

Table 14 shows results across AICrypto categories: Encryptions (ENC), Foundation (FUN), Pseudorandomness (PR), and Signatures (SIGN).

| Model | ZS | CoT | PoT | F-1 |
|---|---|---|---|---|
| *Encryptions (n=3)* | | | | |
| GPT-5 | 97.90 | **99.00** | 88.70 | 96.90 |
| Gemini 2.5 Pro | 94.80 | **100.00** | **100.00** | **100.00** |
| Qwen3-30B | 68.00 | 51.50 | **99.00** | 89.70 |
| Qwen3-235B | 64.40 | **88.70** | 59.30 | 76.50 |
| DeepSeek-V3.1 | 47.40 | 70.60 | **79.20** | 78.40 |
| *Foundation (n=5)* | | | | |
| GPT-5 | 89.60 | **98.60** | 83.30 | **98.60** |
| Gemini 2.5 Pro | 78.40 | 95.70 | 92.80 | **100.00** |
| Qwen3-30B | 77.70 | 65.50 | 78.80 | **91.40** |
| Qwen3-235B | 78.40 | 76.60 | 80.60 | **83.80** |
| DeepSeek-V3.1 | 80.60 | 82.40 | **84.20** | 78.40 |
| *Pseudorandomness (n=7)* | | | | |
| GPT-5 | 78.70 | 97.50 | 95.10 | **100.00** |
| Gemini 2.5 Pro | 65.60 | 72.10 | 73.80 | **82.00** |
| Qwen3-30B | 65.60 | 70.50 | **72.10** | 59.00 |
| Qwen3-235B | 17.20 | 18.00 | 23.00 | **53.30** |
| DeepSeek-V3.1 | 59.00 | 38.50 | 45.90 | **77.00** |
| *Signatures (n=3)* | | | | |
| GPT-5 | **100.00** | **100.00** | **100.00** | **100.00** |
| Gemini 2.5 Pro | 84.80 | **100.00** | **100.00** | **100.00** |
| Qwen3-30B | 98.50 | **100.00** | 21.20 | **100.00** |
| Qwen3-235B | **100.00** | 98.50 | **100.00** | **100.00** |
| DeepSeek-V3.1 | 84.80 | **100.00** | 98.50 | 98.50 |

Table 14: AICrypto breakdown by category (%). Best per row in **bold**.

## E Qualitative Example Details

Figure 2 shows three representative examples where F-1 succeeds while baselines fail. In **Physics**, baselines confused "independent of $x$" with "equals zero"; F-1's explicit formalization of the Lorentz equations prevented this semantic confusion. In **Finance**, all baselines computed the correct value (≈6.3%) but produced incorrect output formats; F-1's formalization step guided decimal form throughout. In **Cryptography**, baselines attempted invalid direct reductions, while F-1 identified the correct sub-hybrid argument by first formalizing the PRF security game. These examples demonstrate that F-1's benefits stem from different mechanisms: preventing semantic confusion, enforcing format consistency, and enabling structured proof construction.

This appendix provides full model responses, judge scores, and feedback for the qualitative examples referenced in Section 6.6.

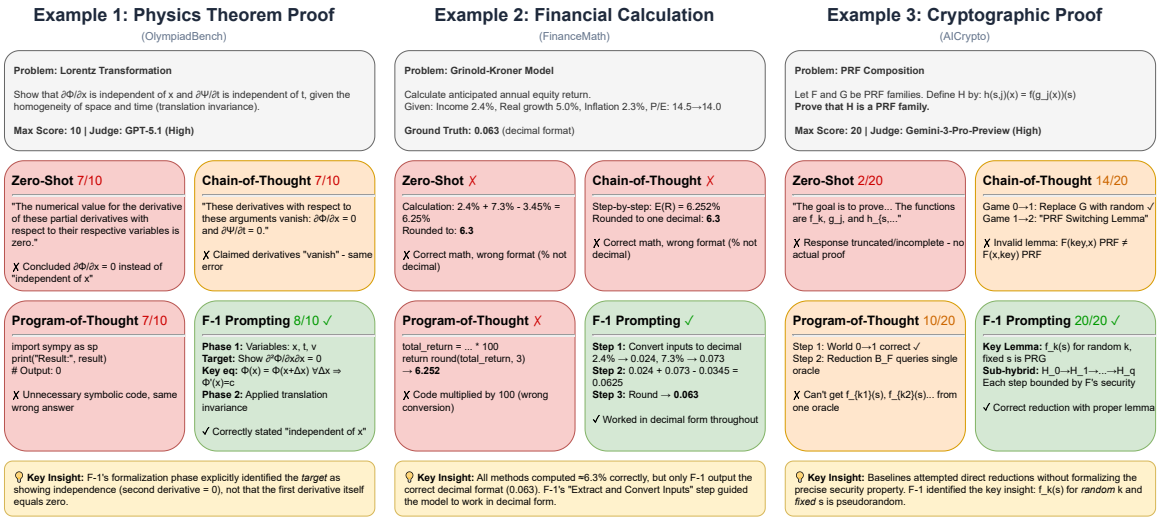


Figure 2: Qualitative comparison. **(Left) Physics:** Baselines confused "independent of $x$" with "equals zero"; F-1's formalization prevented this. **(Center) Finance:** Only F-1 output correct decimal format (0.063). **(Right) Cryptography:** F-1 identified correct sub-hybrid argument while baselines used invalid reductions.

## E.1 Example 1: Lorentz Transformation (OlympiadBench Physics)

**Problem**

Show that $\frac{\partial X}{\partial x}$ is independent of $x$ and $\frac{\partial T}{\partial t}$ is independent of $t$, given the homogeneity of space and time (translation invariance).

**Gold Answer:** Independence (partial derivatives depend only on $v$, not on $x$ or $t$).

**Zero-Shot Response** (Judgment: Partial | Score: 7/10)

**By homogeneity (translation invariance):**

*Spatial homogeneity:* For any shift $a$, $X(x+a,t,v)$ must relate frames the same way as $X(x,t,v)$, up to the corresponding shift of the primed origin. This implies $X(x+a,t,v) - X(a,0,v) = X(x,t,v)$. Differentiate with respect to $a$ at $a=0$ to get $\partial X/\partial x(x,t,v) = \partial X/\partial x(0,0,v)$, i.e., $\partial X/\partial x$ is independent of $x$. Equivalently, $\partial/\partial x(\partial X/\partial x) = 0$.

*Temporal homogeneity:* For any shift $b$, $T(x,t+b,v) - T(0,b,v) = T(x,t,v)$. Differentiate with respect to $b$ at $b=0$ to get $\partial T/\partial t(x,t,v) = \partial T/\partial t(0,0,v)$, i.e., $\partial T/\partial t$ is independent of $t$. Equivalently, $\partial/\partial t(\partial T/\partial t) = 0$.

Thus, the numerical value for the derivative of these partial derivatives with respect to their respective variables is zero.

`So the final answer is \boxed{0}.`

**Zero-Shot Judge Feedback**

**Feedback:** Your use of homogeneity as translation invariance and differentiating with respect to the translation parameter is exactly the right idea and correctly shows that $\partial X/\partial x$ and $\partial T/\partial t$ are independent of $x$ and $t$. The only issue is the final boxed answer: what is constant is these partial derivatives (they can depend on $v$ but not on $x$ or $t$), not that they themselves must equal zero. It would be better to conclude explicitly that $\partial X/\partial x(x,t,v) = \text{constant}(v)$ and $\partial T/\partial t(x,t,v) = \text{constant}(v)$, rather than writing the answer as 0.

**Scores:** Correctness: 7/10 | Reasoning: 8/10 | Completeness: 7/10

**Chain-of-Thought Response** (Judgment: Partial | Score: 7/10)

**1) Understand the problem:**
We are given general transformations $x' = X(x,t,v)$, $t' = T(x,t,v)$ between inertial frames and asked to use homogeneity of space and time to show that $\partial X/\partial x$ is independent of $x$ and $\partial T/\partial t$ is independent of $t$.

**2) Given and to find:**
Given: Homogeneity of space and time (invariance under translations of $x$ and $t$).
Need to show: $\partial X/\partial x$ does not depend on $x$; $\partial T/\partial t$ does not depend on $t$.

**3) Break into steps:**
Express homogeneity via invariance under shifts: $x \to x+a$, $t \to t+b$. Use this to relate $X(x+a,t+b,v)$ to $X(x,t,v)$ up to a constant. Differentiate to see how partial derivatives behave. Conclude independence.

**4) Solve systematically:**
By homogeneity, shifting the origin by $(a,b)$ changes transformed coordinates only by an additive constant:

$$X(x+a,t+b,v) = X(x,t,v) + C_x(a,b,v)$$
$$T(x+a,t+b,v) = T(x,t,v) + C_t(a,b,v)$$

Differentiate first equation w.r.t. $x$: $\partial X/\partial x(x+a,t+b,v) = \partial X/\partial x(x,t,v)$.
Since $a,b$ arbitrary, $\partial X/\partial x$ is invariant under shifts $\Rightarrow$ cannot depend on $x$. Thus $\partial/\partial x(\partial X/\partial x) = 0$.
Similarly for $T$: $\partial/\partial t(\partial T/\partial t) = 0$.

**5) Combine results:**
The homogeneity implies the $x$-dependence of $\partial X/\partial x$ and $t$-dependence of $\partial T/\partial t$ vanish. Numerically: $\partial/\partial x(\partial X/\partial x) = 0$ and $\partial/\partial t(\partial T/\partial t) = 0$.

`So the final answer is \boxed{0}.`

**CoT Judge Feedback**

**Feedback:** You correctly use homogeneity (translation invariance) and the idea of shifting coordinates to argue that the partial derivatives are invariant under shifts, so they cannot depend on $x$ or $t$. The main flaw is in the final statement suggesting the derivatives themselves are zero; what follows from your argument is that their derivatives with respect to $x$ or $t$ vanish, so they are independent of those variables, not that they are numerically zero. Tighten the argument for why translation invariance implies only additive constants, and clarify that the result is independence from $x$ or $t$, not vanishing.

**Scores:** Correctness: 7/10 | Reasoning: 7/10 | Completeness: 8/10

**Program-of-Thought Response** (Judgment: Partial | Score: 7/10)

**Step 1 - Reasoning from homogeneity:**
Homogeneity (translation invariance) of space and time implies: shifting the origins $x \to x + a$ and $t \to t + b$ should not change the functional form of the boost. Hence for any $a, b$,

$$X(x + a, t + b, v) - X(a, b, v) = X(x, t, v).$$

Differentiate both sides with respect to $a$:

$$\frac{\partial}{\partial a}[X(x + a, t + b, v) - X(a, b, v)] = 0$$

which gives $X_x(x + a, t + b, v) - X_x(a, b, v) = 0$ for all $a, b$.
Therefore $X_x$ is independent of $x$ (and $t$), i.e., $\partial X/\partial x$ depends only on $v$.
Similarly, $T_t$ is independent of $t$ (and $x$), i.e., $\partial T/\partial t$ depends only on $v$.

A practical way to encode this structure is that, due to homogeneity, X and T must be affine in $x$ and $t$:

$$X(x, t, v) = A(v)x + B(v)t + C(v)$$
$$T(x, t, v) = D(v)x + E(v)t + F(v)$$

Then $\partial X/\partial x = A(v)$ (independent of $x$), $\partial T/\partial t = E(v)$ (independent of $t$). Equivalently, their second derivatives with respect to $x$ and $t$, respectively, are zero.

**Step 2 - Python code to verify symbolically and extract the numerical result:**
We'll use sympy to model the most general affine form implied by homogeneity, and confirm that the $x$-derivative of $\partial X/\partial x$ and the $t$-derivative of $\partial T/\partial t$ are zero. That zero is the requested numerical value.

```
import sympy as sp

# Symbols
x, t, v = sp.symbols('x t v')

# Coefficients can depend on v, but are independent of x
    and t
A = sp.Function('A')(v)
B = sp.Function('B')(v)
C = sp.Function('C')(v)
D = sp.Function('D')(v)
E = sp.Function('E')(v)
F = sp.Function('F')(v)

# General affine form implied by homogeneity
X = A*x + B*t + C
T = D*x + E*t + F

# First derivatives
dXdx = sp.diff(X, x)    # = A(v), independent of x
dTdt = sp.diff(T, t)    # = E(v), independent of t

# Check independence by differentiating again w.r.t x and
    t
d2Xdx2 = sp.diff(dXdx, x)  # should be 0
d2Tdt2 = sp.diff(dTdt, t)  # should be 0

print("dX/dx =", dXdx)
print("dT/dt =", dTdt)
print("d^2X/dx^2 =", d2Xdx2)   # numerical value 0
print("d^2T/dt^2 =", d2Tdt2)   # numerical value 0

# For the problem's requested numerical output, we
    provide the value 0
result = int(d2Xdx2)
print("Result:", result)
```

**Step 3 - Running the code:**
- $\partial X/\partial x = A(v)$
- $\partial T/\partial t = E(v)$
- $\partial^2 X/\partial x^2 = 0$
- $\partial^2 T/\partial t^2 = 0$
- Result: 0

Therefore, the numerical value that encodes "independence" (the derivative of $\partial X/\partial x$ with respect to $x$, and of $\partial T/\partial t$ with respect to $t$) is 0.

`So the final answer is \boxed{0}.`

**PoT Judge Feedback**

**Feedback:** You correctly used homogeneity (translation invariance) to derive that $\partial X/\partial x$ and $\partial T/\partial t$ cannot depend on $x$ or $t$, which is exactly the structural point the question targets. However, asserting that X and T must be fully affine in $x$ and $t$ is stronger than what follows strictly from this part of the argument, and the symbolic code plus the reported numerical result 0 are unnecessary and somewhat misleading. What matters is that these derivatives are functions of $v$ only, not that their derivatives are numerically zero. Refine the answer by clearly stating: $\partial X/\partial x = f(v)$ and $\partial T/\partial t = g(v)$, with no additional computation or boxed numerical result.

**Scores:** Correctness: 7/10 | Reasoning: 7/10 | Completeness: 6/10

**F-1 Response** (Judgment: Correct | Score: 8/10)

**1) Key variables and target:**
Variables: $x, t, v$
Target: $\partial X/\partial x$ independence of $x$ and $\partial T/\partial t$ independence of $t$. Numerically equivalent to showing $\partial/\partial x(\partial X/\partial x) = 0$ and $\partial/\partial t(\partial T/\partial t) = 0$.

**2) Main equations:**
Definition of partial derivative:

$$\partial X/\partial x(x, t, v) = \lim_{h\to 0}[X(x + h, t, v) - X(x, t, v)]/h$$
$$\partial T/\partial t(x, t, v) = \lim_{k\to 0}[T(x, t + k, v) - T(x, t, v)]/k$$

Homogeneity (translation invariance):
$X(x + \Delta, t, v) - X(x, t, v)$ depends only on $\Delta$ and $v$ (not on $x$ or $t$).
$T(x, t + \Delta, v) - T(x, t, v)$ depends only on $\Delta$ and $v$ (not on $x$ or $t$).

**3) CoT:**
By homogeneity, $X(x+h, t, v) - X(x, t, v) = f(h, v)$, independent of $x$ and $t$.
Hence $\partial X/\partial x(x, t, v) = \lim_{h\to 0} f(h, v)/h = g(v)$,

independent of $x$.
Therefore $\partial/\partial x(\partial X/\partial x) = 0$.
Similarly, $\partial T/\partial t = r(v)$, so $\partial/\partial t(\partial T/\partial t) = 0$.

**4) Conclusion:**
The spatial derivative $\partial X/\partial x$ is independent of $x$; the time derivative $\partial T/\partial t$ is independent of $t$. Numerically: $\partial/\partial x(\partial X/\partial x) = 0$ and $\partial/\partial t(\partial T/\partial t) = 0$.

`So the final answer is \boxed{0}.`

**F-1 Judge Feedback**

**Feedback:** Your use of homogeneity via translation invariance and the limit definition of partial derivatives is solid and essentially complete. To improve, focus the conclusion on the functional independence ($\partial X/\partial x = g(v)$, $\partial T/\partial t = r(v)$) rather than framing it as a 'numerical' statement with boxed zeros, which can confuse the main conceptual point.

**Scores:** Correctness: 8/10 | Reasoning: 7/10 | Completeness: 9/10

**Error Analysis: Lorentz Transformation**

| Method | Judgment | Corr. | Reas. | Comp. |
|---|---|---|---|---|
| Zero-Shot | Partial | 7 | 8 | 7 |
| CoT | Partial | 7 | 7 | 8 |
| PoT | Partial | 7 | 7 | 6 |
| **F-1** | **Correct** | **8** | 7 | **9** |

**Key Finding:** F-1's explicit formalization of the *target* (independence) prevented the conceptual error shared by all baselines.

## E.2 Example 2: Market Analysis & Economics (FinanceMath)

This example demonstrates a critical error pattern: **"Correct Calculation, Wrong Format."** All baseline methods computed the correct numerical value but failed due to output format mismatch (percentage vs. decimal).

**Problem Statement**

An investor from Australia currently manages an equity portfolio valued at A$240 million. He is contemplating adjusting the portfolio based on an evaluation of the risk and return possibilities confronting the Australian economy. Using the data in the table, determine the anticipated annual equity return using the Grinold–Kroner model (the number of outstanding shares is presumed to remain unchanged).
*Given data:* Expected income return: 2.4%, Expected real earnings growth: 5.0%, Expected inflation: 2.3%, Current P/E: 14.5x, Expected P/E: 14.0x
**Ground Truth:** 0.063 (decimal format)

**Zero-Shot Response** (Judgment: Incorrect | Answer: 6.3)

I need to calculate the anticipated annual equity return using the Grinold–Kroner model. The formula is:
Expected Return = Dividend Yield + Earnings Growth + Change in P/E ratio

**Identify components:**
Expected Income Return (dividend yield) = 2.4%
Expected Nominal Earnings Growth = Real growth + Inflation = 5.0% + 2.3% = 7.3%
Change in P/E = $(14.0 - 14.5)/14.5 = -0.5/14.5 \approx -3.448\%$

**Calculation:**
Total = $2.4\% + 7.3\% + (-3.448\%) = 6.252\%$
Rounded to one decimal place (consistent with input data precision): **6.3%**

`Final Answer: 6.3`

**Chain-of-Thought Response** (Judgment: Incorrect | Answer: 6.3)

**Step 1: Identify Grinold–Kroner components**
The model decomposes total return into: (1) Income component (dividend yield), (2) Earnings growth component, (3) Change in valuation (P/E ratio)

**Step 2: Extract data and convert**
Dividend yield = 2.4%
Nominal earnings growth = 5.0% (real) + 2.3% (inflation) = 7.3%
%Δ(P/E) = $(14.0 - 14.5)/14.5 = -3.448\%$

**Step 3: Apply formula**
$E(R) = 2.4\% + 7.3\% - 3.448\% = 6.252\%$
Rounded to one decimal: **6.3%**

`Final Answer: 6.3`

**Program-of-Thought Response** (Judgment: Incorrect | Answer: 6.252)

**Reasoning:** Apply Grinold–Kroner model. Since shares outstanding are unchanged, that term is zero.

```
def solution():
    expected_income_return = 2.4
    current_pe = 14.5
    expected_pe = 14.0
    expected_real_earnings_growth = 5.0
    expected_inflation = 2.3

    pe_change_percent = ((expected_pe - current_pe)
                         / current_pe) * 100
    earnings_growth = expected_real_earnings_growth \
                      + expected_inflation

    total_return = expected_income_return \
                   + pe_change_percent \
                   + earnings_growth
    return round(total_return, 3)

# Result: 6.252
```

`Final Answer: 6.252`

**F-1 Response** (Judgment: Correct | Answer: 0.063)

**1. Problem Formalization**
*Model:* Grinold–Kroner expected equity return
*Givens:* Income return = 2.4%, Real growth = 5.0%, Inflation = 2.3%, P/E: 14.5 → 14.0
*Target:* Expected annual return **as decimal (not percentage)**

**2. Governing Equation**

$$E(R) = \frac{D}{P} + g_{\text{nominal}} + \frac{\Delta(\text{P/E})}{\text{P/E}_0}$$

**3. Step-by-Step Calculation (in decimal)**
Dividend yield: 0.024

Nominal growth: $0.05 + 0.023 = 0.073$
P/E change: $(14.0 - 14.5)/14.5 = -0.034482759$
Total: $0.024 + 0.073 - 0.034482759 = 0.062517241$

**4. Format Verification**
Round to 3 decimals: 0.062517... $\rightarrow$ $\boxed{0.063}$

**Error Pattern: "Correct Calculation, Wrong Format"**

**Observation:** All methods computed ≈6.3% but only F-1 output correct format.

| Method | Computed | Output | Issue |
|---|---|---|---|
| Zero-Shot | 6.3% | 6.3 | % instead of decimal |
| CoT | 6.3% | 6.3 | % instead of decimal |
| PoT | 6.252% | 6.252 | `*100` conversion |
| **F-1** | 6.3% | **0.063** | *Correct format* |

**Root Cause:** Baselines work in percentage throughout without considering output format.
**F-1 Advantage:** Formalization step ("Extract and Convert Inputs") naturally guides model to work in decimal form, matching expected output format.

## E.3 Example 3: Cryptographic Proof (AICrypto/ProofBench)

This example demonstrates F-1's advantage in **cryptographic security proofs**. The problem requires proving that a composed function family is a Pseudorandom Function (PRF), evaluated by an LLM judge on mathematical rigor.

**Problem Statement**

Let $F = \{f_k^n : \{0,1\}^n \rightarrow \{0,1\}^n\}$ and $G = \{g_j^n : \{0,1\}^n \rightarrow \{0,1\}^n\}$ be PRF families. Define the function family $H$ by:

$$h_{s,j}^n(x) = f_{g_j^n(x)}^n(s)$$

where the key is $(s, j) \in \{0,1\}^n \times \{0,1\}^n$. **Prove that $H$ is a PRF family.**
*Category:* Foundations (FUN) *Max Score:* 20 points
*Judge:* Gemini 3 Pro Preview

**Zero-Shot Response** (Score: 2/20)

**Analysis:**
The goal is to prove that $H$ defined by $h_{s,j}(x) = f_{g_j(x)}(s)$ is a PRF family given that $F$ and $G$ are PRF families.
Let's denote the security parameter by $n$. The functions are $f_k$, $g_j$, and $h_{s,j}$...
*[Response truncated/incomplete]*

**Zero-Shot Judge Feedback**

This submission is unfortunately **incomplete**. You have restated the problem statement and defined the notation, but the proof cuts off immediately after the setup. There is no actual proof, reduction, or analysis provided. I cannot evaluate the mathematical correctness or rigor of an argument that is not present.

**Chain-of-Thought Response** (Score: 14/20)

**Proof Strategy:** Hybrid argument with two games.
**Game 0 $\rightarrow$ Game 1:** Replace $g_j^n$ with random function $\rho$. Reduction to $G$'s security. ✓
**Game 1 $\rightarrow$ Game 2:** Replace $f_{\rho(x)}^n(s)$ with random function.
*Approach:* Use "PRF Switching Lemma," claiming that if $F$ is a PRF, then the dual family $F'$ defined by $f'_s(k) = f_k(s)$ is also a PRF.
*[Applies reduction based on this lemma...]*

**CoT Judge Feedback**

Your proof structure is logical. However, there is a **significant flaw** in Game 2 regarding the "PRF Switching Lemma". You claim that if $F$ is a PRF, its dual $F'$ (swapping key and input) is also a PRF. **This is false generally.** A PRF is only guaranteed to be pseudorandom with respect to the input, not the key. For example, if the first bit of the key leaks, $F_k(x)$ might still be a PRF, but $F_x(k)$ would not be. This invalidates the reduction in Game 2.

**Program-of-Thought Response** (Score: 10/20)

**Step 1 (World 0 $\rightarrow$ World 1):** Replace inner PRF $G$ with random function.
Reduction to $G$'s security: *correct*. ✓
**Step 2 (World 1 $\rightarrow$ World 2):** Oracle is $f_{\rho(x)}^n(s)$ where $\rho$ is random.
Claim: Distinguishing $f_{k_i}(s)$ (random keys, fixed input) from random relies on PRF security of $F$.
*[Constructs reduction $\mathcal{B}_F$ that queries single oracle $O_F$...]*

**PoT Judge Feedback**

Step 1 is correct. However, **Step 2 is incorrect**. PRF security guarantees indistinguishability of $f_k(x_i)$ (fixed key, varying inputs) from random. Here, you have $f_{k_i}(s)$ (varying keys, fixed input).
Your reduction fails: $\mathcal{B}_F$ gets oracle $O_F$ which is either $f_k(\cdot)$ or $R(\cdot)$. It queries $O_F(s)$. But you need $f_{k_1}(s), f_{k_2}(s), \ldots$ for *many different* keys $k_i$. Your reduction only has access to *one* key $k$. You cannot generate samples for multiple keys using a single oracle.

**F-1 Response** (Score: 20/20)

**1. Goal & Security Assumptions**
Prove $\mathsf{Adv}_{H,\mathcal{A}}^{\mathsf{PRF}}(n)$ is negligible for any PPT $\mathcal{A}$.
**2. Hybrid Argument Structure**
*Game 0:* Oracle $O_0(x) = f_{g_j^n(x)}^n(s)$ *Game 1:* Oracle $O_1(x) = f_{r_G(x)}^n(s)$ *Game 2:* Oracle $O_2(x) = r(x)$
**3. Game 0 $\rightarrow$ Game 1:** Standard reduction to $G$'s PRF security. $|P_0 - P_1| \leq \mathsf{negl}(n)$ ✓
**4. Key Lemma for Game 1 $\rightarrow$ Game 2:**
*"If $F$ is a PRF, then for any fixed $s$, the distribution $\{f_k^n(s)\}_{k \leftarrow \{0,1\}^n}$ is computationally indistinguishable from uniform."*
*Proof:* Build $\mathcal{B}_F$ that queries its oracle at single point $s$. If oracle is $f_k^n$, output is from $\{f_k^n(s)\}$; if random function, output is uniform.
**5. Sub-Hybrid Argument:** Define $H_0, \ldots, H_q$ where $H_i$ answers first $i$ queries with random values. Each $|P(H_{i-1}) - P(H_i)|$ bounded by $F$'s security. Sum over $q$ steps remains negligible.
**Conclusion:** $|P_0 - P_2| \leq |P_0 - P_1| + |P_1 - P_2| \leq \mathsf{negl}(n)$ $\therefore H$ is a PRF family. □

**F-1 Judge Feedback**

This is an **excellent, textbook-quality proof**. The hybrid argument is clearly structured, and the reduction strategies for both steps are handled correctly. You correctly identified that replacing $G$ with a random function results in independent keys for $F$. The "swap" logic is implicitly handled correctly by noting that we rely on the pseudorandomness of $F$ with respect to a random key, even though the input $s$ is fixed. Your sub-hybrid argument for the $F$-transition is rigorous and correctly bounds the advantage. **Well done.**

**Error Pattern: Incorrect Security Reductions**

**Observation:** All baselines failed to correctly handle the "varying keys, fixed input" scenario.

| Method | Score | Critical Flaw |
|---|---|---|
| Zero-Shot | 2/20 | Response incomplete (truncated) |
| CoT | 14/20 | Invalid "PRF Switching Lemma" assumption |
| PoT | 10/20 | Reduction uses single oracle for multiple keys |
| **F-1** | **20/20** | *Correct sub-hybrid with key lemma* |

**Root Cause:** Baselines attempted direct reductions without formalizing the precise security property needed.
**F-1 Advantage:** Formalization identified the key insight: $f_k^n(s)$ for *random* $k$ and *fixed* $s$ is pseudorandom, enabling a valid sub-hybrid argument.

# F Evaluation Prompts

This appendix provides the complete evaluation prompts used for each benchmark. We use two evaluation paradigms: (1) **rule-based scoring** with deterministic regex-based extraction for benchmarks with definitive numerical answers, and (2) **LLM-as-Judge** for proof-based benchmarks requiring semantic evaluation.

## F.1 Rule-Based Scoring Benchmarks

**Code Execution** For PoT baselines and F-1's PoT strategy, generated code is executed in a sandboxed environment (30s timeout, standard libraries). When F-1 selects PoT as its solving strategy, the generated code is executed identically to the baseline, with the output used as the final answer.

**FinanceMath & OlympiadBench** These benchmarks have definitive numerical answers. We extract the final answer using regex patterns matching `\boxed{}` or the last numerical value, then compare against ground truth with tolerance $\epsilon = 10^{-6}$ for floating-point answers.

**IMO-AnswerBench** For IMO problems with definitive answers, we use the deterministic autograder from Luong et al. (2025):

**IMO-AnswerBench Auto-Grader Prompt**

**System Role:** Deterministic Mathematical Autograder

You are a precise, automated grading system. Your sole function is to determine if the final answer provided in the Model Solution is mathematically equivalent to the Golden Answer. You must NOT grade the reasoning or steps, only the final result.
**Equivalence Rules:**
- **Algebraic:** e.g., 'n(n+1)/2' ≡ 'n^2/2 + n/2'
- **Numerical:** e.g., '1/2' ≡ '0.5'; 'sqrt(2)/2' ≡ '1/sqrt(2)'
- **Set/List:** Order does not matter unless specified as ordered tuple
- **No Partial Credit:** Incomplete or partially incorrect → Incorrect

**Output:** \boxed{Correct} or \boxed{Incorrect}

## F.2 LLM-as-Judge Benchmarks

For proof-based benchmarks where correctness requires semantic understanding, we employ LLM-as-Judge evaluation with the best-performing models available for each domain: Gemini-3-Pro (High) for IMO-ProofBench and AICrypto, GPT-5.1 (High) for OlympiadBench TP.

**IMO-ProofBench** Mathematical proofs are evaluated on a 0–7 scale following Luong et al. (2025):

**IMO-ProofBench Judge Prompt**

You are an expert grader for the International Mathematics Olympiad (IMO). Evaluate the proposed solution strictly and rigorously.
**Scoring Rubric (0–7 scale):**
- **7 Points (Correct):** Complete, correct, and fully rigorous
- **6 Points (Almost):** Sound core argument with minor errors/gaps
- **1 Point (Partial):** Substantial progress on key steps
- **0 Points (Incorrect):** No substantial progress or fundamentally flawed

**Evaluation Process:**
1. Analyze problem and ground truth solution
2. Step-by-step verification of every logical step
3. Identify all flaws, gaps, and errors
4. Compare against grading guidelines

**Output:** `<points>N out of 7</points>`

**AICrypto (CryptoProof)** Cryptographic proofs are evaluated on correctness, completeness, and rigor:

**AICrypto Judge Prompt**

**System:** You are a professor of theoretical computer science specializing in mathematical foundations of cryptography, grading student homework for CS 6857 (Graduate Cryptography).
**Grading Rubric (100 points, scaled to problem max):**
- **Mathematical Correctness (60%):** Sound reasoning, justified claims, correct reductions, accurate probability analysis
- **Completeness (25%):** All cases covered, no logical gaps, edge cases handled
- **Rigor (15%):** Formal and precise, correct use of definitions, consistent notation

**Common Issues:**
1. Incorrect use of cryptographic definitions (confusing OWF with PRF)
2. Missing verification of critical conditions
3. Intuitive arguments without formal justification
4. Logical gaps in reduction proofs
5. Incorrect probability analysis

**Output:** JSON with score, feedback, key_errors, strengths

## F.3 Evaluation Summary

Table 15 summarizes the evaluation methodology for each benchmark.

| Benchmark | Method | Scale | Threshold |
|---|---|---|---|
| FinanceMath | Rule-Based (regex) | Binary | Exact match |
| OlympiadBench OE | Rule-Based (regex) | Binary | $\epsilon = 10^{-6}$ |
| OlympiadBench TP | LLM Judge | Binary | Correct/Incorrect |
| IMO-AnswerBench | Rule-Based (LLM) | Binary | Equivalent |
| IMO-ProofBench | LLM Judge | 0–7 | $\geq 6$ = Correct |
| AICrypto | LLM Judge | 0–100 | $\geq 80\%$ = Correct |

Table 15: Evaluation methodology summary. LLM Judge uses Gemini-3-Pro (High) for IMO-ProofBench and AICrypto, GPT-5.1 (High) for OlympiadBench TP.

## G Variant Accuracy and Statistical Tests

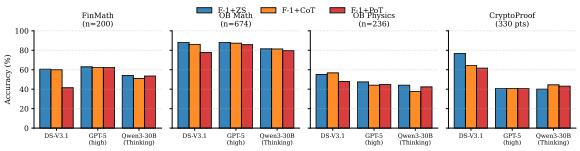


Figure 3: Visual comparison of F-1 variant accuracy at matched maximum reasoning effort. Same data as Table 16.

| Model | Variant | FinMath | OB Math | OB Phys | Crypto |
|---|---|---|---|---|---|
| DeepSeek-V3.1 | F-1+ZS | **60.50** | **88.13** | 55.08 | **76.8** |
| | F-1+CoT | 60.00 | 86.05 | **56.78** | 64.5 |
| | F-1+PoT | 41.50 | 77.74 | 47.88 | 61.6 |
| GPT-5 (high) | F-1+ZS | **63.00** | **88.13** | **47.46** | **40.9** |
| | F-1+CoT | 62.50 | 87.24 | 44.07 | 40.8 |
| | F-1+PoT | 62.50 | 85.76 | 44.92 | **40.9** |
| Qwen3-30B-Thk | F-1+ZS | **54.00** | **81.45** | **44.07** | 40.0 |
| | F-1+CoT | 51.00 | 81.31 | 37.71 | **44.5** |
| | F-1+PoT | 53.50 | 79.67 | 42.37 | 43.1 |

Table 16: Accuracy (%) across F-1 variants at matched maximum reasoning effort. F-1+ZS is the strongest or tied-strongest variant on 9 of 12 (model$\times$benchmark) cells. F-1+PoT is the weakest variant on math-heavy benchmarks for DeepSeek-V3.1 and GPT-5.

Table 17 reports pairwise statistical comparisons across the three F-1 variants on OlympiadBench OE benchmarks for which per-problem correctness is recorded.

| Bench / Comparison | Model | $\Delta$% | 95% CI | p |
|---|---|---|---|---|
| *OB Math (n=674)* | | | | |
| F-1+ZS vs F-1+PoT | DeepSeek | +10.39 | [+7.42, +13.50] | **<0.001** |
| F-1+CoT vs F-1+PoT | DeepSeek | +8.31 | [+5.49, +11.28] | **<0.001** |
| F-1+ZS vs F-1+CoT | DeepSeek | +2.08 | [−0.15, +4.30] | 0.087 |
| F-1+ZS vs F-1+PoT | GPT-5 | +2.37 | [+0.30, +4.60] | **0.044** |
| F-1+ZS vs F-1+PoT | Qwen3 | +1.78 | [−0.45, +4.01] | 0.162 |
| *OB Physics (n=236)* | | | | |
| F-1+ZS vs F-1+PoT | DeepSeek | +7.20 | [+1.27, +13.14] | **0.024** |
| F-1+CoT vs F-1+PoT | DeepSeek | +8.90 | [+2.97, +14.83] | **0.005** |
| F-1+ZS vs F-1+CoT | Qwen3 | +6.36 | [+1.27, +11.02] | **0.020** |
| F-1+ZS vs F-1+CoT | GPT-5 | +3.39 | [−0.85, +7.63] | 0.185 |

Table 17: Pairwise statistical tests on OlympiadBench OE for F-1 variants. McNemar's exact binomial $p$-values; bootstrap 95% CIs from 10,000 resamples. Bold marks $p < 0.05$.

## H Strategy Distribution Analysis

To understand *how* F-1 adapts its solving behavior across domains, we classify the strategy used in each F-1 response based on output characteristics (presence of code blocks, step markers, and formula density). Table 18 and Figure 4 present the distribution across benchmarks, averaged over all models.

| Benchmark | N | CoT | PoT | Direct | Hybrid | F-ratio |
|---|---|---|---|---|---|---|
| FinanceMath | 1,000 | 45.3 | 31.4 | 2.3 | 21.0 | 28.2 |
| IMO-Bench | 2,300 | 28.0 | 4.3 | 2.4 | 65.3 | 29.8 |
| OlympiadBench | 5,752 | 43.0 | 4.9 | 0.5 | 51.6 | 26.5 |
| AICrypto | 90 | 61.1 | 0.0 | 1.1 | 37.8 | 15.9 |

Table 18: F-1 strategy distribution (%) across benchmarks, pooled over 5 models. **F-ratio**: average formula density in responses. F-1 strongly favors PoT on computation-heavy FinanceMath (31.4%) while selecting CoT/Hybrid for proof-oriented AICrypto and IMO-Bench. Per-model breakdowns are in Section I.

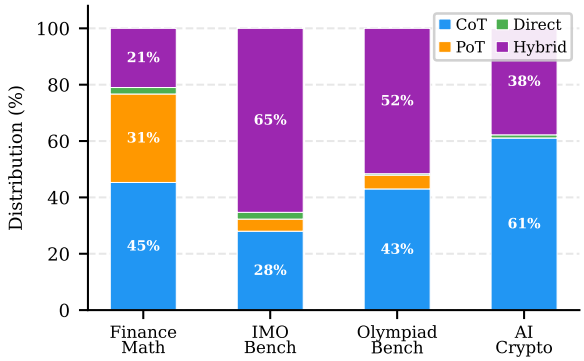


Figure 4: F-1 strategy distribution across domains (pooled over 5 models). Without explicit routing, F-1 naturally favors PoT for computation-heavy FinanceMath, Hybrid for competition math (IMO-Bench), and CoT for proof-oriented AICrypto.

The distribution reveals clear domain adaptation: FinanceMath has the highest PoT usage (31.4%), reflecting numerical computation demands; AICrypto uses exclusively CoT and Hybrid strategies (0% PoT), consistent with proof problems; and IMO-Bench shows the highest Hybrid usage (65.3%). Notably, these distributions emerge without explicit routing rules, reflecting each model's internal assessment of equation structure.

## I Per-Model Strategy Distribution

Table 19 provides per-model strategy distributions for F-1 across all benchmarks.

| Benchmark | Model | CoT | PoT | Direct | Hybrid |
|---|---|---|---|---|---|
| FinanceMath | GPT-5 | 0.0 | 92.0 | 6.0 | 2.0 |
| | Gemini 2.5 Pro | 28.0 | 44.5 | 0.0 | 27.5 |
| | Qwen3-30B | 69.5 | 0.0 | 0.0 | 30.5 |
| | Qwen3-235B | 78.0 | 0.0 | 2.0 | 20.0 |
| | DeepSeek-V3.1 | 51.0 | 20.5 | 3.5 | 25.0 |
| IMO-Bench | GPT-5 | 36.1 | 21.1 | 10.4 | 32.4 |
| | Gemini 2.5 Pro | 18.5 | 0.2 | 1.7 | 79.6 |
| | Qwen3-30B | 19.3 | 0.0 | 0.0 | 80.7 |
| | Qwen3-235B | 20.0 | 0.0 | 0.0 | 80.0 |
| | DeepSeek-V3.1 | 45.9 | 0.0 | 0.0 | 54.1 |
| OlympiadBench | GPT-5 | 36.1 | 15.4 | 0.8 | 47.8 |
| | Gemini 2.5 Pro | 33.0 | 1.5 | 1.2 | 64.3 |
| | Qwen3-30B | 47.6 | 0.0 | 0.0 | 52.4 |
| | DeepSeek-V3.1 | 55.2 | 2.8 | 0.0 | 42.0 |
| AICrypto | GPT-5 | 72.2 | 0.0 | 5.6 | 22.2 |
| | Gemini 2.5 Pro | 33.3 | 0.0 | 0.0 | 66.7 |
| | Qwen3-30B | 66.7 | 0.0 | 0.0 | 33.3 |
| | Qwen3-235B | 55.6 | 0.0 | 0.0 | 44.4 |
| | DeepSeek-V3.1 | 77.8 | 0.0 | 0.0 | 22.2 |

Table 19: Per-model F-1 strategy distribution (%). GPT-5 heavily favors PoT on FinanceMath (92%) while Qwen models prefer CoT/Hybrid. On IMO-Bench, Qwen and Gemini models predominantly use Hybrid (∼80%), while GPT-5 distributes more evenly. Note: OlympiadBench is missing Qwen3-235B data.

## J Efficiency, Upper Bound, and Win/Tie/Loss

| Benchmark | All× | All✓ | Adapt✓ | Adapt× | F-1 Only |
|---|---|---|---|---|---|
| IMO-Bench | 90.7 | 4.8 | 2.5 | 1.8 | 0.3 |
| OlympiadBench | 33.5 | 31.5 | 21.3 | 10.5 | 3.1 |
| FinanceMath | 33.0 | 19.6 | 29.1 | 12.8 | 5.5 |
| AICrypto† | 18.9 | 37.8 | 21.1 | 14.4 | 7.8 |

Table 20: Per-problem outcome analysis (%, averaged across 5 models). **All×/All✓**: all four methods fail/succeed (uninformative for strategy choice). **Adapt✓**: F-1 and at least one baseline succeed (F-1 picks a working strategy). **Adapt×**: at least one baseline succeeds but F-1 fails. **F-1 Only**: only F-1 succeeds. †$n$=18, interpret with caution.

| Method | Efficiency Ratio ↑ FinMath | Olympiad | IMO | Crypto | Avg | Overhead |
|---|---|---|---|---|---|---|
| Zero-Shot | 1.97 | 0.97 | 0.48 | 1.38 | 1.20 | — |
| CoT | 2.29 | 0.99 | 0.47 | 1.31 | 1.26 | +35 tok |
| PoT | 2.51 | 0.70 | **0.52** | 1.22 | 1.24 | +64 tok |
| **F-1 (Ours)** | **3.04** | **1.08** | 0.49 | **1.42** | **1.51** | +68 tok |

Table 21: Token efficiency (Efficiency Ratio = Accuracy/Tokens × 100) and prompt overhead vs. Zero-Shot. F-1 achieves the highest efficiency with minimal overhead (+68 tokens). All methods use a single API call.

| Benchmark | Upper Bound | F-1 | F-1 / UB |
|---|---|---|---|
| FinanceMath | 67.0% | 54.2% | 80.9% |
| OlympiadBench | 66.5% | 55.9% | 84.1% |
| AICrypto† | 81.1% | 66.7% | 82.2% |

Table 22: Upper bound (UB) on applied domains, following analysis in Self-Consistency (Wang et al., 2023b) and ToT (Yao et al., 2023). Upper Bound = 100% − All Failed represents the best achievable by *any single baseline*. F-1 = All✓ + Adapt✓ + F-1 Only (from Table 20). †n=18, interpret with caution.

| Model | vs Zero-Shot W | T | L | vs CoT W | T | L | vs PoT W | T | L |
|---|---|---|---|---|---|---|---|---|---|
| GPT-5 | 3 | 0 | 1 | 3 | 0 | 1 | 4 | 0 | 0 |
| Gemini | 4 | 0 | 0 | 4 | 0 | 0 | 4 | 0 | 0 |
| Qwen3-30B | 4 | 0 | 0 | 3 | 0 | 1 | 4 | 0 | 0 |
| Qwen3-235B | 3 | 0 | 1 | 4 | 0 | 0 | 3 | 0 | 1 |
| DeepSeek | 3 | 0 | 1 | 3 | 0 | 1 | 4 | 0 | 0 |
| **Total** | 17 | 0 | 3 | 17 | 0 | 3 | 19 | 0 | 1 |

Table 23: Win/Tie/Loss counts for F-1 vs each baseline across 4 benchmarks per model (20 comparisons per baseline). F-1 wins 53/60 total comparisons (88.3%).

## K Weighted Average Results

Table 3 reports macro-averaged accuracy (unweighted by dataset size) to prevent larger benchmarks from dominating. For completeness, Table 24 provides weighted averages, where each benchmark contributes proportionally to its size (IMO-Bench: 460, OlympiadBench: 1,438, FinanceMath: 200, AICrypto: 18).

| Method | Macro Avg | Weighted Avg |
|---|---|---|
| Zero-Shot | 49.50 | 46.13 |
| CoT | 55.30 | 51.86 |
| PoT | 52.64 | 42.11 |
| **F-1 (Ours)** | **61.07** | **54.54** |

Table 24: Macro vs weighted average accuracy (%). F-1 leads in both metrics. The gap between macro and weighted is smaller for F-1, indicating consistent performance across benchmark sizes.

## L Pretraining Pattern Analysis: Supplementary Figures

This appendix provides the full infini-gram-mini analysis supporting Section 3. We query eight pretraining corpora (∼81.7T tokens): DCLM-baseline (16.7T) and seven Common Crawl snapshots (CC-2025-05 through CC-2025-30). Patterns are classified into three tiers: T1 prompt-derived, T2 shared math/reasoning language (including LaTeX notation), and T3 broad proxies. Figure 5 summarizes the DCLM findings.

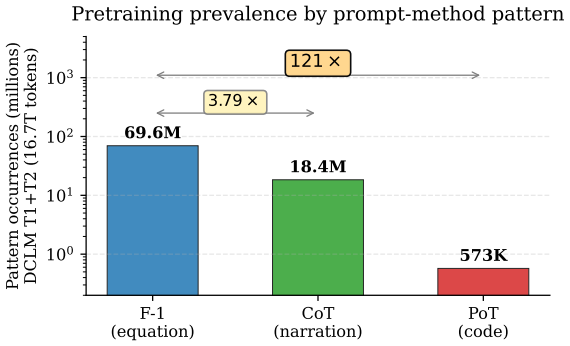


Figure 5: Pretraining prevalence by prompt-method pattern (DCLM-baseline T1+T2 occurrences, log scale). Regenerated from raw infini-gram counts. Equation language (F-1) appears 3.79× as often as step-by-step narration (CoT) and 121× as often as code (PoT), a distributional pattern consistent with the observed F-1 vs. PoT gap on applied math (correlational, not causal).

**Pattern lists.** Table 25 enumerates the representative patterns queried per method, grouped by tier (T1 prompt-derived, T2 shared math/code language, T3 broad proxies).

| Method | Tier | Representative patterns |
|---|---|---|
| CoT | T1 | "step by step", "step-by-step", "think step by step", "solve step by step" |
| | T2 | "first, we need to", "let's start by", "let's break this down", "therefore, the answer is" |
| | T3 | "next, we" |
| PoT | T1 | "write code to", "write a program to", "write Python code to solve" |
| | T2 | `def solve(`, `def solution(`, `return answer`, `print(answer)` |
| | T3 | `import numpy`, `import math`, `import sympy` |
| FP / F-1 | T1 narrative | "governing equation", "key variables", "key equations", "identify givens and target" |
| | T1 LaTeX | `\begin{equation}`, `\begin{align}`, plus starred and end variants |
| | T2 narrative | "solve for", "the formula for", "the equation is", "using the formula", "we need to find" |
| | T2 LaTeX | `\frac{`, `\sqrt{`, `\sum_{`, `\int_{`, `\text{`, `\cdot`, `\leq`, `\geq`, `\implies`, `\equiv` |

Table 25: Method-associated patterns queried in Section 3. Tier assignment validated empirically via n-gram analysis of actual model outputs: T1 ratios exceed 10× between own-method and other-method outputs; T2 ratios fall in 0.7–1.7×; T3 ratios drop below 1×. The FP/F-1 tiers split into *narrative* (English phrases from the prompt) and LaTeX (markup tokens the prompt asks the model to emit).

**LaTeX pattern inclusion.** The F-1 prompt explicitly instructs models to "write equations in LaTeX" (see prompt templates in Section B); we therefore include LaTeX typography patterns alongside narrative phrases. `\frac{` alone contributes 23.9M occurrences on DCLM, the single largest F-1 pattern.

**Methodology caveats.** Raw occurrence counts reflect token-level prevalence, not document-level coverage. A single equation-heavy document may contain `\frac{` 10+ times while "step by step" typically appears once per document. F-1/CoT ratios should therefore be interpreted as upper bounds on document-level prevalence. Nonetheless, the F-1/PoT ratio (121.6×) is robust under any reasonable document normalization, as `\frac{` alone appears in more documents than all PoT T1+T2 patterns combined.

## M F-1 Pattern Blending Analysis

This appendix provides the full output n-gram analysis supporting Section 5.4. We extract model outputs from Qwen3-30B-Thinking-2507 on RQ1 (F-1+Zero-shot) and RQ2 (F-1+CoT, F-1+PoT) variants, covering all four benchmarks (n=1,456 problems per variant). Pattern counts are case-insensitive string matches; per-output document rates are computed as the fraction of outputs containing at least one occurrence of a given pattern.

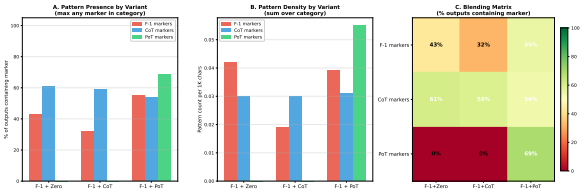


Figure 6: F-1 pattern blending on Qwen3-30B-Thinking ($n$=1,456 per variant). **Left:** F-1 markers persist in 43/32/55% of F-1+Zero/F-1+CoT/F-1+PoT outputs – F-1 preprocessing survives any solving strategy. **Center:** Pattern density per 1K chars. **Right:** Blending matrix – strategy-specific markers cleanly isolate (PoT: 69% in F-1+PoT vs 0% otherwise). "Step by step" appears in 54–61% regardless of variant, confirming it is generic English rather than CoT-specific.

**Why this validates composability.** If F-1 merely replaced CoT/PoT with equation reasoning, we would expect F-1 markers to crowd out strategy-specific markers. Instead, F-1+PoT outputs contain BOTH F-1 equation markers (55%) AND PoT code markers (69%), while F-1+CoT contains both

F-1 equation markers (32%) and natural CoT language. The clean isolation of PoT markers at 0% in non-PoT variants confirms the model respects strategy cues while retaining the F-1 preprocessing phase. This is consistent with F-1 acting as a structural preprocessing prefix that adds equation reasoning without displacing strategy-specific behavior, although a fully causal account would require mechanistic interpretability.

## N Statistical Tests: Detailed Breakdown

Table 3 in the main text reports the overall accuracy results with significance markers. This appendix provides (i) the per-benchmark breakdown of paired tests, (ii) the per-cell McNemar/Wilcoxon results that complement the bolded best-cells in Table 3, and (iii) AICrypto-specific bootstrap CIs and a power analysis. The analysis code and full per-model results are available in the accompanying repository.

| **Benchmark** | **vs ZS** $\Delta$ / $p$ | **vs CoT** $\Delta$ / $p$ | **vs PoT** $\Delta$ / $p$ |
|---|---|---|---|
| Olympiad | +9.71 / <.001 | +2.32 / <.001 | +7.37 / <.001 |
| IMO-Answer | −0.90 / 0.34 | −1.05 / 0.27 | +7.95 / <.001 |
| **Overall** | **+7.39 / <.001** | **+1.58 / <.001** | **+7.50 / <.001** |

Table 26: Per-benchmark breakdown of paired statistical tests. $\Delta$ in pp; full 95% CIs in repository. On OlympiadBench, F-1 beats all baselines; on IMO-AnswerBench, F-1 ties CoT ($p$=0.27, dominated by difficulty ceiling) but beats PoT by +7.95 pp. Pooling both benchmarks retains significance against CoT (Overall row).

**Per-cell McNemar / Wilcoxon tests.** Table 27 reports paired tests for every (benchmark, model) cell of Table 3. We use McNemar on per-problem correctness for IMO-Bench and OlympiadBench, parse-recovered correctness (round-to-3-decimals equality) for FinanceMath, and Wilcoxon signed-rank on partial-credit scores (0–100, $n$=18) for AICrypto.

**AICrypto bootstrap CIs and power analysis.** We defend the $n$=18 AICrypto results without removing them: (i) partial-credit per-problem scores (continuous, 0–100) instead of the binary $\geq 80\%$ threshold; (ii) Wilcoxon signed-rank tests with bootstrap CIs (Table 28); (iii) an explicit power analysis (below).

**Power at $n$=18.** For a paired $t$-test at $\alpha$=0.05, 80% power, the minimum detectable effect (MDE) given the observed within-pair SD of F-1−CoT differences is 19.14 pp at the median model and 24.87 pp at the noisiest (Cohen's $d_z$=0.701). Per-model effects fall below the MDE, which is why per-cell tests are mostly non-significant. Pooling the 90 paired scores across 5 models is the correct level of inference for AICrypto and recovers significance against all three baselines.

## O Failure-Cause Classification: Multi-Judge Cross-Validation

This appendix supplements §5.6. We sampled $n$=88 F-1 failure cases stratified by (benchmark, model): 20 each from FinanceMath, OlympiadBench OE-maths, OE-physics, TP-maths, plus 8 TP-physics (smaller pool); split between GPT-5 and Gemini-2.5-Pro F-1 outputs. Each case was independently classified by three judges of different model lineages – GPT-5.1 (Azure, low reasoning effort), DeepSeek-V4-Pro thinking, and Claude Opus 4.7 – using an identical JSON-output prompt. Multi-judge cross-validation defends against single-judge / self-favoring bias (Manakul et al., 2023). Two cases failed to parse on DeepSeek (the other two judges parsed all 88), leaving $n$=86 where all three judges produced a valid label.

| **Cat.** | **GPT-5.1** | **DS-V4-Pro** | **Opus 4.7** | **Maj.** |
|---|---|---|---|---|
| A | 29 (33.0%) | 22 (25.0%) | 28 (31.8%) | 22 (25.6%) |
| B | 2 (2.3%) | 0 (0.0%) | 3 (3.4%) | 1 (1.2%) |
| C | 57 (64.8%) | 64 (72.7%) | 57 (64.8%) | 63 (73.3%) |
| *Unp.* | 0 | 2 | 0 | — |

Table 29: Per-judge marginal classifications of $n$=88 F-1 failures. **A** = wrong-eq → wrong-ans (over-reliance); **B** = wrong-eq → recovered (resilience); **C** = right-eq → exec-err (mechanical). All three judges converge: most failures (64–73%) preserve a correct Phase-1 equation; 25–33% are over-reliance; self-correction (B) is rare (0–3%). Majority-vote (right column, $n$=86 where all three judges parsed) ties on category C dominance.

**Inter-rater agreement.** Pairwise Cohen's $\kappa$: GPT-5.1 vs DeepSeek-V4-Pro = 0.522 (moderate); GPT-5.1 vs Claude Opus 4.7 = 0.306 (fair); DeepSeek vs Claude = 0.385 (fair). Three-way Fleiss' $\kappa$=0.403 (moderate, $n$=86); 52 of 86 cases (60.5% [95% CI: 49.9, 70.1]) had unanimous agreement across all three judges. While individual case-level agreement is fair-to-moderate, the marginal distributions converge tightly, supporting the high-level claim that Phase-1 formalization is reliable on ∼73% of failure cases regardless of which judge is used.

| Benchmark | Model | Δ vs ZS / $p$ | Δ vs CoT / $p$ | Δ vs PoT / $p$ |
|---|---|---|---|---|
| *IMO-Bench (n=400, McNemar)* | | | | |
| | GPT-5 | −3.25 / 0.182 | −3.25 / 0.182 | +15.00 / <0.001*** |
| | Gemini-2.5-Pro | +1.25 / 0.630 | +0.50 / 0.907 | +9.25 / <0.001*** |
| | Qwen3-30B | −0.25 / 1.000 | +0.50 / 0.897 | +6.75 / 0.003** |
| | Qwen3-235B | −0.50 / 0.839 | −1.50 / 0.286 | −8.25 / <0.001*** |
| | DeepSeek-V3.1 | −1.75 / 0.500 | −1.50 / 0.610 | +17.00 / <0.001*** |
| *OlympiadBench (n=1,438, McNemar)* | | | | |
| | GPT-5 | +2.09 / 0.043* | +2.02 / 0.052 | +13.84 / <0.001*** |
| | Gemini-2.5-Pro | +25.87 / <0.001*** | +1.39 / 0.143 | +5.63 / <0.001*** |
| | Qwen3-30B | +9.94 / <0.001*** | +4.10 / <0.001*** | +5.08 / <0.001*** |
| | Qwen3-235B | +9.79 / <0.001*** | +5.39 / <0.001*** | +7.52 / <0.001*** |
| | DeepSeek-V3.1 | +0.83 / 0.430 | −1.25 / 0.205 | +4.80 / <0.001*** |
| *FinanceMath (n=200, McNemar on parse-recovered correctness)* | | | | |
| | GPT-5 | +38.00 / <0.001*** | +21.50 / <0.001*** | +10.00 / 0.003** |
| | Gemini-2.5-Pro | +24.00 / <0.001*** | +0.50 / 1.000 | +1.00 / 0.845 |
| | Qwen3-30B | +27.50 / <0.001*** | +3.50 / 0.230 | +2.50 / 0.359 |
| | Qwen3-235B | +25.50 / <0.001*** | +26.00 / <0.001*** | +1.00 / 0.855 |
| | DeepSeek-V3.1 | +16.50 / <0.001*** | +16.00 / <0.001*** | +7.00 / 0.088 |
| *AICrypto (n=18 paired, Wilcoxon on 0–100 partial-credit scores)* | | | | |
| | GPT-5 | +7.72 / 0.150 | −0.11 / 0.655 | +8.91 / 0.128 |
| | Gemini-2.5-Pro | +15.65 / 0.043* | +3.25 / 0.180 | +4.10 / 0.180 |
| | Qwen3-30B | +9.75 / 0.063 | +18.27 / 0.051 | +9.23 / 0.444 |
| | Qwen3-235B | +8.97 / 0.255 | +4.91 / 0.530 | +8.36 / 0.289 |
| | DeepSeek-V3.1 | +10.37 / 0.211 | +5.92 / 0.495 | +3.17 / 0.780 |

Table 27: Per-cell paired tests for Table 3. Δ = F-1 minus baseline (pp). $*p<0.05$, $**p<0.01$, $***p<0.001$, two-sided. F-1 vs ZS: 10/20 cells significant (all 5 FinMath, 4/5 Olympiad, 1/5 AICrypto); F-1 vs CoT: 5/20 (3/5 FinMath: GPT-5, Qwen3-235B, DeepSeek; 2/5 Olympiad: Qwen3 family); F-1 vs PoT: 11/20 (5/5 IMO-Bench, 5/5 Olympiad, 1/5 FinMath, 0/5 AICrypto; note Qwen3-235B IMO is significant in the negative direction at $-8.25$ pp). AICrypto per-cell tests are underpowered at $n$=18; pooled across models, F-1 vs CoT $+6.45$ pp ($p$=0.022); see Table 28.

| | vs ZS | | vs CoT | | vs PoT | |
|---|---|---|---|---|---|---|
| **Model** | Δ | $p$ | Δ | $p$ | Δ | $p$ |
| DeepSeek-V3.1 | +10.4 | .211 | +5.9 | .495 | +3.2 | .780 |
| GPT-5 | +7.7 | .150 | −0.1 | .655 | +8.9 | .128 |
| Gemini-2.5-Pro | +15.7 | .043* | +3.3 | .180 | +4.1 | .180 |
| Qwen3-235B | +9.0 | .255 | +4.9 | .530 | +8.4 | .289 |
| Qwen3-30B | +9.8 | .063 | +18.3 | .051 | +9.2 | .444 |
| **Pooled** | **+10.49** | **< .001*** ** | **+6.45** | **.022*** | **+6.75** | **.029*** |

Table 28: AICrypto paired tests on partial-credit scores (per-problem mean, 0–100; per-model $n$=18, pooled $n$=90). Δ in pp, F-1 minus baseline. Wilcoxon signed-rank, two-sided. Per-model tests are underpowered at $n$=18; pooling 5 models restores significance against all three baselines.

**What this defends.** Single-judge classification (especially when the judge belongs to the same family as the model under test) risks self-favoring bias. By using three judges of independent lineages – and reporting the worst-case pairwise $\kappa$ alongside the majority-vote rate – we make the dominant finding (C ≫ A ≫ B) robust to any single judge's idiosyncrasies.

## P Phase-1 Verification Probe

This appendix supplements the Limitations discussion on Phase-1 over-reliance (§7, item 1). The failure-cause analysis (Section O) finds that 25.6% of F-1 failures are wrong-equation → wrong-answer, with only 1.2% self-correcting. A natural mitigation is to add an explicit Phase-1 self-verification step. We test whether such a step recovers a measurable fraction of these failures *within a single call*.

**Variant.** We define F-1+Verify by inserting a Phase 1.5 instruction between Phase 1 and Phase 2:

**F-1+Verify Phase 1.5 instruction**

```
Phase 1.5 - Verify:
Re-read the problem and check whether the
equations correctly represent every given, the
target, and any constraints. If you find a
mismatch, sign error, missing variable, or
wrong governing equation, revise the equations
below before continuing. If correct, state
"Equations verified." and proceed.
```

System prompt and Phase 1/Phase 2 instructions are identical to FP/F-1+ZS on FinanceMath. F-1+Verify therefore differs from FP *only* in the addition of Phase 1.5.

**Setup.** FinanceMath validation ($n$=200, rule-based scoring, $\epsilon$=$10^{-6}$). Model: GPT-5 (high reasoning) – the cell with the largest F-1 lift over CoT in Table 3 and the clearest deployment of the Phase-1 over-reliance failure mode. Both conditions use the same dataset order and decoding settings.

| Method | Acc. | Net flips | Δ | Tokens | Δtok |
|---|---|---|---|---|---|
| F-1 (FP) | 64.00 | — | | 2,243 | — |
| F-1+Verify | 64.50 | +1 | +0.50 | 2,890 | +647 |

Table 30: F-1+Verify vs F-1 on FinanceMath × GPT-5 ($n$=200, paired). **Acc.** = accuracy (%). **Net flips** = (only-F-1+Verify-correct) − (only-F-1-correct) = $9 - 8$. **Tokens** = mean completion tokens per problem. McNemar exact two-sided $p$=1.0.

**Results.** F-1+Verify recovers 9 problems that F-1 misses but breaks 8 that F-1 solves correctly, for a net gain of 1 problem out of 200 (+0.50 pp, McNemar exact $p$=1.0). The probe also incurs a substantial token cost: +647 mean completion tokens per problem (+28.8%) and +115 mean prompt tokens. Crucially, in 200/200 outputs the model concludes Phase 1.5 with "Equations verified." or a textually equivalent acknowledgement; no output revises its Phase 1 equations during Phase 1.5. The behavior the verification step is designed to elicit – self-detection and correction of a wrong governing equation – never fires.

**Interpretation.** The probe converges with the failure-cause finding (Section O) that self-correction is rare (B=1.2% in the multi-judge analysis, 0% here): once the model has committed to a governing equation in Phase 1, explicit single-call self-verification does not surface the error. The over-reliance is structural, not addressable by adding a verification turn within the same generation. Multi-call backtracking (Yao et al., 2023; Liu et al., 2023) – where Phase 1 outputs are evaluated by a separate call before solving – remains the more promising mitigation, at the cost of leaving the single-call regime that F-1 targets.

**Scope.** This probe runs on a single (benchmark, model) cell with the largest expected gain and the cleanest failure-mode signal. Negative findings here do not preclude the possibility that a different verification prompt formulation, a different benchmark, or a smaller / non-reasoning model could yield different behavior. The result establishes that the most direct single-call mitigation is not effective for the cell that should most benefit from it.

## Q F-1 vs Formulate-and-Solve: Per-Problem Complementarity

This appendix supplements the F-1 vs F&S discussion in §2. We re-implemented Formulate-and-Solve following Kao et al. (2024) – a single-call prompt that asks the model to (i) formulate the problem as a system of equations over named variables, then (ii) solve algebraically – and ran it on FinanceMath ($n$=200) and OlympiadBench Math (OE_TO_maths_en_COMP, $n$=674) with two of our paper's models: GPT-5 (proprietary, frontier) and Qwen3-30B-A3B-Thinking-2507 (open, mid-tier). Table 31 cross-tabulates per-problem correctness in all four (model, benchmark) cells.

| Bench | Model | N | F-1 | F&S | F1\FS | FS\F1 | Union |
|---|---|---|---|---|---|---|---|
| FinMath | GPT-5 | 200 | 64.00 | 64.00 | 5.5 | 5.5 | 69.5 |
| FinMath | Qwen30B | 200 | **56.00** | 51.50 | 9.5 | 5.0 | 61.0 |
| OB Math | GPT-5 | 674 | 86.35 | **87.09** | 3.4 | 4.2 | 90.5 |
| OB Math | Qwen30B | 674 | 78.64 | 78.64 | 7.9 | 7.9 | 86.5 |

Table 31: F-1 vs Formulate-and-Solve (Kao et al., 2024) per-problem cross-tabulation. **F-1 / F&S** columns: aggregate accuracy (%). **F1\FS / FS\F1**: % of problems solved only by F-1 / only by F&S (set-difference). **Union** = (both+F1-only+FS-only)/$N$. Aggregate accuracies are close (within ±5 pp) but the methods solve substantially different problem subsets: 7.6–15.7% disjoint across cells. Combining both methods exceeds either alone by +4–8 pp, confirming F-1's adaptive Phase 2 is orthogonal to F&S's fixed algebraic-solve pipeline. Best per-cell aggregate accuracy in **bold**.

**Reading the table.** On FinanceMath, F-1 wins on Qwen3-30B by +4.5 pp and ties GPT-5; on OB Math, F&S wins on GPT-5 by +0.74 pp and ties Qwen3-30B. None of the per-cell aggregate gaps are large. *However*, the disjoint-success columns reveal that even when accuracies tie, the two methods are solving different problems: e.g., on Qwen3-30B OB Math both methods score 530/674 but 53 problems are solved only by F-1 and 53 only by F&S – a 15.7% disjoint set. This is direct evidence that F-1's contribution is qualitatively different from F&S's, not a marginal accuracy improvement on the same problem distribution. The union (both methods combined as an oracle) reaches 86.5–90.5% on OB Math and 61–70% on FinanceMath,

indicating substantial unrealized headroom from method ensembling.

**Why aggregate accuracies tie despite disjoint success sets.** The per-cell ties (e.g., Qwen3-30B OB Math both at $530/674$) are not coincidence: each method solves a comparable number of problems, but the *identities* of those problems differ. F-1 tends to win on multi-step problems requiring strategy switching mid-solution – where its Phase 2 menu (CoT/PoT/Direct) lets the model defer the solving-mode choice until after equation formalization. F&S, with its fixed algebraic-solve pipeline, tends to win on problems where a clean equation system is the bottleneck and downstream solving is mechanical. The two failure modes are largely orthogonal, which is why aggregate counts converge while individual problems diverge.

**Implication for method ensembling.** The union accuracies suggest a simple downstream ensemble: run both F-1 and F&S, accept the answer when they agree, defer to a verifier when they disagree. On Qwen3-30B OB Math, this would bound test-time accuracy at $86.50\%$ (the union) – a $+7.86$ pp lift over either method alone. We do not pursue ensembling here because our focus is single-call methods under fixed compute, but it is a promising direction for follow-up work targeting applied-math benchmarks where governing equations exist but solution structure varies.

## R BBH Negative Control: Per-Strategy Details

This appendix supplements §6.7. We applied F-1 verbatim (Phase-1 formalization + Phase-2 strategy menu) to BBH word_sorting (Suzgun et al., 2023) – a task that requires alphabetically sorting a list of 5–10 English words and contains no equations or quantitative structure. We ran three strategies: **Zero-Shot**, **CoT** ("think step by step"), and **F-1** (full prefix). Models tested: GPT-5 (Azure), Gemini-2.5-Pro, and Qwen3-30B-A3B-Thinking-2507 served via vLLM. Each cell evaluates 250 problems.

**Why F-1 fails on word_sorting.** The Phase-1 directive ("identify governing equations, write them in LaTeX") has no semantic referent for a list of words. Inspection of GPT-5 F-1 outputs shows the model attempting to (i) cast sorting as an algebraic permutation problem with formal index variables, (ii) write LaTeX expressions like $\sigma : \{w_1, \ldots, w_n\} \to \{w_{\sigma(1)}, \ldots, w_{\sigma(n)}\}$, and (iii) "solve" for the permutation – a route that produces consistent answer-format errors and occasional sorting mistakes. The instruction-following models obey the formalization directive literally; the thinking-mode model partially overrides it during its internal trace, which is why Qwen3-30B-Thinking regresses substantially less.

| Model | ZS | CoT | F-1 |
|---|---|---|---|
| GPT-5 | 98.8 (247) | 96.0 (240) | **46.8 (117)** |
| Gemini-2.5-Pro | 87.6 (219) | 98.4 (246) | **56.0 (140)** |
| Qwen3-30B-Th. | 94.0 (235) | 93.2 (233) | 88.4 (221) |

Table 32: BBH word_sorting full results across 3 models and 3 strategies (each cell: accuracy% (correct/250)). F-1 regression cells in bold for GPT-5 and Gemini ($-49.2$ and $-42.4$ pp vs CoT). Qwen3-30B-Thinking regresses only $-4.8$ pp, reflecting that thinking-mode models partially adapt away from inappropriate prompt instructions during their internal reasoning trace.

**Implication.** This cleanly delineates F-1's deployment envelope: F-1 is a targeted intervention for problems with identifiable governing equations (applied math, physics, finance, engineering), *not* a generic prompting upgrade. We recommend practitioners restrict F-1 to domains where Phase 1's instruction has a meaningful semantic referent.

## S Document-Level Pattern Normalization

This appendix supplements the robustness check in §3. The token-level infini-gram counts (Section L, Figure 5) report total occurrences of method-associated phrases. A reviewer concern is that LaTeX typography commands (e.g., `\frac{`) can appear multiple times per equation-heavy document, so token-level prevalence may overstate document-level coverage. We address this by sampling random occurrences per pattern and counting the number of unique source documents.

**Sampling protocol.** For each pattern, we drew $K{=}100$ random occurrence locations via the infini-gram `find` endpoint and resolved each to a source document ID via the `get_doc_by_rank` endpoint. The unique-document count $U$ over $K$ samples gives an estimate of average occurrences per document, $\widehat{occ/doc} \approx K/U$. Combined with the token total $T$, the estimated number of distinct documents containing the pattern is $\widehat{D} = T \cdot U/K$. We restrict the sampling to DCLM-baseline (the curated, highest-quality corpus) and to a representative subset of T1+T2 patterns (10 CoT, 10 PoT,

21 F-1 patterns covering both narrative phrases and LaTeX environments).

| Method | Tokens | Est. Docs | Tok/Doc |
|---|---|---|---|
| CoT | 18,829,256 | 18,826,446 | 1.00 |
| PoT | 1,183,071 | 1,179,270 | 1.00 |
| F-1 | 10,755,424 | 10,739,381 | 1.00 |

Table 33: Document-level normalization of infini-gram pattern counts on DCLM-baseline. Estimated occurrences per document are ≈1.0 across all three methods, indicating that token-level ratios are not artifactually inflated by repeat-occurrence concentration in a small set of documents.

**Result.** Table 33 reports the aggregate token counts, estimated unique document counts, and tokens-per-document for each method on DCLM. The tokens-per-document ratio is ≈1.0 for all three methods (CoT 1.0001, PoT 1.0032, F-1 1.0015), so each pattern occurrence is, on average, in its own distinct document. This holds even for high-volume LaTeX patterns: `\begin{equation}` (451K tokens, 451K docs), `\begin{align}` (467K tokens, 462K docs). The two patterns where occ/-doc deviates from 1.0 ("key governing equations" at 8.88, "equations in latex" at 1.67) have very small token totals (71 and 886 respectively) and contribute negligibly to the aggregate.

**Implication.** The token-level F-1/CoT and F-1/PoT ratios reported in §3 closely track their document-level counterparts – the document-coverage caveat does not materially change the ranking. F-1 patterns are genuinely (not artifactually) rarer than CoT in pretraining; the gain mechanism is structural ordering of equation-first prompting at inference, not a corpus-statistical artifact.

## T AI Usage Statement

LLMs were used solely to assist with writing tasks, such as proofreading, grammatical corrections, and LaTeX formatting. All research ideas, primary content, and initial drafts were created by the authors, and LLMs did not generate the core manuscript material. LLMs were also used to support code generation and debugging during evaluation and analysis. All resulting content are carefully reviewed and validated by the authors.