Before you <think>, monitor: Implementing Flavell's metacognitive framework in LLMs

Nick Oh
socius labs
London, UK
nick.sh.oh@socius.org

Abstract

Current approaches to enhancing LLM reasoning follows two isolated paradigms: *Monitor-Generate* methods like Plan-and-Solve (Wang et al., 2023) and SELF-DISCOVER (Zhou et al., 2024) excel at strategic planning but lack mechanisms to verify whether selected strategies succeed; while *Generate-Verify* approaches like Self-Verification (Weng et al., 2022) and SELF-REFINE (Madaan et al., 2023) iteratively refine outputs but commence generation blindly without task assessment. This separation creates inefficiencies – strategies fail without feedback, and refinement occurs without strategic grounding. We address this gap by implementing Flavell's cognitive monitoring model (1979) from the broader *Monitor-Generate-Verify* framework (Oh & Gobet, 2025), operationalising it as a three-phase iterative system. On GSM8K, preliminary results show 75.42% accuracy versus 68.44% for SELF-REFINE and 67.07% for Self-Verification, while requiring fewer attempts (1.3 vs 2.0) at 27-37% increased inference cost. These initial findings suggest upfront monitoring produces higher-quality initial solutions that reduce refinement needs, though evaluation beyond arithmetic reasoning is needed to establish generalisability.

1 Introduction

This preliminary experimental report documents our initial implementation and testing of Flavell's (1979) cognitive monitoring model for enhancing LLM. We operationalise the *Monitor-Generate-Verify* framework proposed by Oh & Gobet (2025), creating a system where models explicitly assess task difficulty before attempting solutions, adapt their computational resources accordingly, and evaluate outputs along structured metacognitive dimensions. Our experiments on a subset of GSM8K problems compare this approach against Self-Verification and SELF-REFINE baselines using `Llama-3.1-8B-Instruct`. While our evaluation is limited to 659 arithmetic problems with a single model, the results suggest that explicit monitoring may reduce the need for extensive iterative refinement by producing better initial solution attempts. We present these findings as an early exploration of whether cognitive science principles, particularly metacognitive monitoring, can inform more effective reasoning architectures for language models.

This paper is structured as follows: Section 2 situates our approach within existing paradigms. Section 3 presents our implementation of Flavell's cognitive monitoring model. Section 4 evaluates performance against baselines. Section 5 discusses limitations and future directions. Section 6 reflects on the methodological implications of our approach.

2 Related Works

2.1 Monitor-Generate (MG) Methods

Monitor-Generate (MG) approaches recognise that effective reasoning requires understanding task structure before attempting solutions. *Zero-shot prompting methods* have evolved from simple planning to sophisticated strategy discovery: Plan-and-Solve (Wang et al., 2023) replaces generic "think step by step" with structured guidance for problem decomposition and careful execution, achieving 76.7% accuracy on arithmetic tasks; SELF-DISCOVER (Zhou et al., 2024) enables LLMs to identify task-intrinsic reasoning structures through meta-prompts (SELECT, ADAPT, IMPLEMENT), yielding 27-32% improvements on reasoning tasks; and Meta-Reasoning Prompting (Gao et al., 2024) functions as an adaptive router, evaluating multiple reasoning methods against task requirements to select the optimal approach. *Few-shot prompting methods* leverage exemplars for enhanced strategy selection, with Strategic Chain-of-Thought (Wang et al., 2024) first eliciting strategic knowledge then applying it systematically (achieving 21.05% improvement on GSM8K), and HYBRIDMIND (Han et al., 2024) employing a meta-selector to choose between natural language, symbolic, or hybrid reasoning. *Reinforcement learning based approaches* like Elastic Reasoning (Scalable Chain-of-Thought) (Xu et al., 2025) optimise strategy through trained budget allocation between thinking and solution phases. Despite their sophistication in pre-generation planning, these methods lack mechanisms to verify whether selected strategies actually succeed or to learn from failed attempts.

2.2 Generate-Verify (GV) Methods

Generate-Verify (GV) approaches focus on iterative refinement through self-evaluation. Self-Verification (Weng et al., 2022) generates multiple candidates then validates through backward reasoning, systematically masking conditions to reconstruct original problems and selecting based on aggregated verification scores. SELF-REFINE (Madaan et al., 2023) implements a three-phase cycle where the same model serves as generator, critic, and refiner, maintaining complete iteration history to prevent repeating mistakes, achieving 20% average improvement without additional training. Recent advances emphasise sophisticated verification mechanisms: ReVISE (Lee et al., 2025) enables intrinsic self-verification through structured curriculum learning with confidence-aware decoding, while theoretical work formalises self-improvement through the 'generation-verification gap' that scales with model pre-training FLOPs (Song et al., 2024). The pivotal role of verifier quality has been demonstrated by Zhang et al. (2024), showing substantial gains with strong verifiers but limitations with weak self-verifiers. However, even strong verification cannot fully overcome the 'prefix dominance trap' (Luo et al., 2025): once models begin with suboptimal reasoning strategies, performance degrades by nearly 20% with minimal recovery through subsequent verification. This exposes a fundamental limitation – these approaches commence generation without assessing task characteristics or retrieving problem-solving strategies, leading to inefficient exploration that initial strategic planning could avoid.

3 `Monitor-Generate-Verify` Framework

The preceding analysis reveals a fundamental gap. MG methods excel at strategic planning but cannot verify success, while GV approaches refine outputs iteratively but lack initial task assessment. Addressing these complementary limitations, we implement the *Monitor-Generate-Verify* (MGV) framework (Oh & Gobet, 2025) – specifically operationalising Flavell's cognitive monitoring model (1979) as a three-phase iterative reasoning system. Rather than operating within a single paradigm, MGV synthesises both: it monitors task characteristics to inform strategy selection (addressing GV's blind generation), executes with adaptive parameters, then verifies outcomes to guide subsequent cycles (addressing MG's lack of systematic validation). This creates a complete metacognitive loop where verification informs monitoring, monitoring guides generation, and generation produces solutions whose evaluation feeds back to refine initial assessments.

The framework operates through T cycles, where each cycle $\tau \in \{0, 1, ..., T-1\}$ consists of three phases. Our implementation employs *zero-shot prompting* throughout, avoiding the need for task-specific examples while testing the generality of metacognitive principles (complete prompts are provided in Appendix A). Following Flavell's assumption of pre-existing metacognitive knowledge, we adopt the 20 problem-solving strategies compiled by Didolkar et al. (2024), who instructed LLMs to assign skill labels to GSM8K problems.

While this knowledge base was constructed using `gpt-4-0613`, raising questions about cross-model transferability – akin to humans having privileged access to their own metacognition (Nelson & Narens, 1990) and findings that LLMs better predict their own behaviour than other models' (Binder et al., 2024) – we adopt it as a reasonable starting point for this preliminary investigation. This knowledge base primarily represents $\mathcal{MK}_{\text{Strategy}}$ from Flavell's three categories ($\mathcal{MK}_{\text{Agent}}$, $\mathcal{MK}_{\text{Task}}$, $\mathcal{MK}_{\text{Strategy}}$), focusing on problem-solving approaches rather than agent capabilities or task characteristics.

Algorithm 1 `Flavell's Model of Cognitive Monitoring`

Require: task $\mathcal{T}$, model M, strategies $\mathcal{MK}$, prompts $\mathcal{P} = \{p_{monitor}, p_{strategy}, p_{execute}, p_{verify}\}$

Initialize $\mathcal{S}_0 = \text{ACTIVE}$, $\tau = 0$, $\mathcal{ME}^{-1}_{\text{evaluative}} = \emptyset$
while $\mathcal{S}_\tau = \text{ACTIVE}$ and $\tau < T$ **do**
  // Monitor: Assess task difficulty
  $\mathcal{ME}^{\tau}_{\text{difficulty}}, features_\tau = M(p_{mon} \| \mathcal{T} \| \mathcal{ME}^{\tau-1}_{\text{evaluative}})$ ▷ Assess task difficulty

  // Generate: Apply cognitive strategy
  $strategy_\tau = M(p_{str} \| features_\tau \| \mathcal{ME}^{\tau}_{\text{difficulty}} \| \mathcal{MK})$ ▷ Choose approach
  $solution_\tau = M(p_{exe} \| \mathcal{T} \| strategy_\tau)$ ▷ Execute strategy with adaptive params

  // Verify: Evaluate performance
  $\mathcal{ME}^{\tau}_{\text{evaluative}} = M(p_{ver} \| \mathcal{T} \| solution_\tau)$ ▷ Evaluate output quality

  $\mathcal{S}_{\tau+1} =$ **if** $\text{mean}(\mathcal{ME}^{\tau}_{\text{evaluative}}) \geq 0.85$ **then** TERMINATE **else** ACTIVE
  $\tau = \tau + 1$
end while
return $y_{\arg\max_i \text{mean}(\mathcal{ME}^{i}_{\text{evaluative}})}$

The following subsections detail each component of our implementation, examining how monitoring assess task difficulty, how generation employs dual-stage processing, and how verification provides structured metacognitive feedback.

3.1 Monitor

The model analyses problems without solving them, identifying task characteristics and assessing difficulty on a [0,1] scale (Appendix A.1). While Steyvers & Peters (2025) propose *implicit elicitation* of metacognitive states through token likelihood measurements, we employ *explicit elicitation*, prompting the model to verbalise its difficulty assessment. For $t > 0$, the monitor receives evaluation scores from the previous cycle $(coherence, plausibility, consistency, goal\_conduciveness) \in [0,1]^4$, enabling metacognitive recalibration – lower scores trigger upward difficulty adjustment, which subsequently increases computational resources during generation.

This abstraction mechanism, where detailed failure information is compressed into difficulty reassessment, speculatively parallels human metacognition: rather than explicitly recalling every step of failed attempts, we often carry forward an implicit sense that "this is harder than I thought" that shapes subsequent approaches. Setting aside the cognitive parallel, this compression offers a computationally efficient alternative to maintaining complete episodic traces, encoding failure patterns directly into resource allocation decisions.

3.2 Generate

Generation employs a two-stage approach: strategy selection from 20 domain-specific approaches based on monitored features and difficulty, followed by solution execution with adaptive parameters (Appendices A.2). Harder problems receive proportionally more resources – token budgets scale as $400 + \mathcal{ME}_{\text{difficulty}} \times 400$ and temperature as $0.3 + \mathcal{ME}_{\text{difficulty}} \times 0.2$ – reflecting that harder problems may benefit from expanded exploration while maintaining sufficient determinism for reliable computation. For $t > 0$, generation receives complete previous cycle context (attempted strategy, previous reasoning, diagnostic feedback), preventing redundant approaches and enabling targeted improvements.

The cognitive plausibility of separating selection from execution is well-established. Reder (1987) identifies a distinct "strategy selection phase" in human question answering preceding execution, while dual strategy models (Beeson et al., 2019) emphasise selection as a separate cognitive act. Empirical evidence reinforces this distinction. Rickard (2004) demonstrates that practice transitions people from multi-step algorithmic strategies to direct retrieval, highlighting selection-execution independence. Research on arithmetic problem-solving further shows these functions are independently affected by cognitive load (Imbo et al., 2007).

3.3 Verify

Verification evaluates solutions along four Flavellian dimensions – coherence (logical connectivity), plausibility (approach reasonableness), consistency (computational accuracy), and goal-conduciveness (question answering) – producing both numerical scores $\in [0, 1]^4$ and diagnostic text explaining specific strengths or failures (Appendix A.3). The framework terminates when mean evaluation score ≥ 0.85 or after T cycles, implementing a satisficing strategy that accepts good-enough solutions rather than pursuing optimality. This structured evaluation distinguishes strategy selection errors (requiring different approaches) from execution errors (requiring careful reapplication), informing the monitoring phase's subsequent recalibration and creating the complete metacognitive loop.

4 Experimental Results

We evaluate our implementation of Flavell's MGV model against Self-Verification and SELF-REFINE baselines (Appendix B) on 659 randomly sampled problems from the GSM8K test set (Cobbe et al., 2021). Comparing across this spectrum from simple verification of multiple candidate solutions to iterative refinement offers preliminary insights into whether gains stem from monitoring or merely additional computation.

4.1 Experimental Setup

All experiments utilised `Llama-3.1-8B-Instruct` with task-specific zero-shot prompt configurations on an NVIDIA H100 SXM GPU. For MGV, we implement Flavell's framework with $T = 3$ maximum cycles, terminating early if the mean evaluation score exceeds 0.85. The Monitor phase assesses difficulty $\in [0, 1]$, which dynamically adjusts generation parameters: token budget scales as $400 + 400 \times \mathcal{ME}_{\text{difficulty}}$ and temperature as $0.3 + 0.2 \times \mathcal{ME}_{\text{difficulty}}$. Strategy selection draws from 20 predefined approaches compiled by Didolkar et al. (2024). The Verify phase evaluates solutions along four dimensions (coherence, plausibility, consistency, goal-conduciveness), each scored on [0,1] with 300 tokens allocated for evaluation.

Self-Verification generates 3 candidate solutions at temperature 0.3 (800 tokens each), then validates through backward verification with majority voting at temperature 0.3. SELF-REFINE iterates through generate-critique-refine cycles with temperature 0.3 (800 tokens), terminating after 3 cycles or upon receiving positive feedback.

4.2 Results

Table 1 presents comparative results across accuracy, computational cost, and solution attempts. Here, "attempts" indicate reasoning cycles before termination: for MGV, each attempt is one Monitor-Generate-Verify cycle; for SELF-REFINE, each attempt is one generate-critique-refine iteration; for Self-Verification, attempts represent candidate generation and verification rounds using majority voting.

Method	Accuracy	Avg Time (s)	Avg Attempts
Self-Verification	442/659 (67.07%)	7.52	1.2
Self-REFINE	451/659 (68.44%)	6.98	2.0
MGV (Flavell)	**497/659 (75.42%)**	9.60	1.3

Table 1: Performance comparison on GSM8K (659 problems)

Flavell's MGV model achieves 75.42% accuracy on GSM8K, a 7-8 percentage point improvement over both baselines and representing a 22% relative error reduction compared to SELF-REFINE. Notably, this gain is achieved with fewer average attempts than SELF-REFINE (1.3 vs 2.0), with approximately 70% of problems solved on the first cycle – suggesting that metacognitive monitoring produces higher-quality initial solutions that reduce the need for extensive iteration. This efficiency in attempt count tentatively validates that explicit monitoring helps avoid the prefix dominance trap (Luo et al., 2025), where poor initial strategies cascade into unrecoverable errors. However, these benefits incur computational overhead: MGV requires 27.7% more time than Self-Verification and 37.5% more than SELF-REFINE, adding approximately 2-3 seconds per problem due to the monitoring and strategy selection phases. This trade-off – higher accuracy with fewer attempts but increased inference time – positions MGV as particularly suitable for applications prioritising solution quality over real-time constraints, while potentially prohibitive for latency-critical deployments.

5 Limitations and Future Directions

Beyond the immediate limitations of evaluating solely on arithmetic reasoning and lacking multiple-run statistical validation, our work exposes several key questions.

Probing Metacognitive Monitoring A fundamental limitation lies in our reliance on *explicit elicitation* (Steyvers & Peters, 2025) – prompting models to verbalise their metacognitive experience of difficulty. These outputs depend on the model's ability to represent and articulate internal states in language, which may not accurately reflect actual computational processes. Studies consistently find that implicit confidence measures derived from token likelihoods exhibit greater metacognitive sensitivity than explicitly prompted confidence (Xiong et al., 2023), highlighting a gap between what models internally represent and what they express. This discrepancy is particularly concerning given Lindsey et al. (2025)'s demonstration that while `Claude-3.5-Haiku` accurately reports intermediate steps, it hallucinates non-existent computational processes when describing its internal mechanisms for simple addition, despite correctly activating relevant neural pathways. More promising directions involve implicit estimation of metacognitive states. Research reveals that features like confidence and certainty correspond to linearly separable directions in representation space (Zou et al., 2023; Liu et al., 2023). Notably, Ji-An et al. (2025) identify a low-dimensional 'metacognitive space' within LLMs' neural representations, suggesting models monitor a subset of their mechanisms – opening possibilities for direct neural probes of metacognitive experiences including Feeling-of-Knowing, Ease-of-Learning, and Judgements-of-Learning in future MGV implementations.

Modular Architecture and Verification Challenges Our uniform 8B model overlooks differential capability requirements across phases. Qin et al. (2025) decompose mathematical reasoning into Plan, Execute, and Verify – strikingly parallel to MGV framework – revealing that small models (0.5B) reliably identify solution plans while execution demands substantially larger models, suggesting potential efficiency gains from size-differentiated architectures. However, verification poses the greatest challenge. Stechly et al. (2024) demonstrate self-critique consistently degrades performance through false negatives, hallucinated feedback, and insensitivity to critique levels, with improvements emerging only through external sound verifiers. Notably, performance gains prove independent of critique richness – multiple attempts with reliable verification suffices – aligning with Zhang et al. (2024)'s findings on verifier criticality and explaining Self-Verification's efficiency (1.2 attempts through candidate generation). MGV's design – abstracting feedback into difficulty rather than specific corrections, and switching strategies rather than editing solutions – may reduce error propagation, though self-verification remains fundamentally unreliable. Future implementations should explore modular architectures: smaller monitoring models (3B), standard generation models (8B), and either substantially larger models or external symbolic verifiers, creating an LLM-Modulo framework (Kambhampati et al., 2024) leveraging specialised capabilities while maintaining metacognitive coherence.

Inherent Reasoning Boundaries Qin et al. (2025) observe that execution failures often stem from models applying steps based on spurious correlations – superficial formatting or phrasing patterns – rather than robust reasoning, echoing the prefix dominance trap (Luo et al., 2025). While reinforcement learning improves execution by reducing low-level errors, it cannot expand fundamental mathematical understanding, only optimising what models already implicitly know. This suggests MGV shares inherent limits in addressing genuinely novel problems beyond the model's latent capabilities. The framework enhances navigation of known solution spaces but cannot transcend the underlying model's reasoning boundaries.

Metacognitive Knowledge and Adaptation Adopting `gpt-4-0613`'s strategies (Didolkar et al., 2024) for `Llama-3.1-8B` neglects potential model-specific metacognitive representations, paralleling how humans possess privileged access to their own metacognition (Nelson & Narens, 1990). Future work should explore learning model-intrinsic metacognitive knowledge through self-supervised methods. The Monitor component could leverage V-STaR's approach (Hosseini et al., 2024), which utilises both successful and failed reasoning attempts – training via preference learning to distinguish effective from ineffective strategies rather than discarding failures. Applied to MGV, this would enable the Monitor to learn from strategy selection errors: contrasting high-scoring solutions against low-scoring ones to progressively refine difficulty assessments and strategy mappings. Through iterative bootstrapping across reasoning cycles, the system could discover model-specific problem taxonomies and emergent strategies beyond those available in external knowledge bases, transforming MGV from imposing borrowed metacognitive structure to cultivating genuine model-intrinsic metacognitive capabilities.

6 Discussion

MGV is not technically "novel" in its individual computational mechanisms. Its monitoring phase – assessing task difficulty and selecting strategies – parallels the pre-generation analysis found in MG paradigm. And its generate-verify loop with iterative refinement mirrors GV paradigm. What might seem distinctive – combining upfront strategic planning with iterative refinement – can be approximated by sequentially applying existing methods. Nevertheless, the four evaluation dimensions (coherence, plausibility, consistency, goal-conduciveness), while structured, function as evaluation metrics rather than fundamental algorithmic innovations. From a purely technical perspective, MGV appears to be a well-engineered integration of established techniques.

Yet the distinction lies not in technical superiority but in methodological origins. While MG and GV methods draw *inspiration* from human reasoning processes, using cognitive insights to guide engineering decisions, MGV tests whether formal psychological theories can directly *translate* into computational systems. This theory-first implementation's preliminary success opens an underexplored research direction: established cognitive theories can become testable machine learning hypotheses. This approach could potentially accelerate progress by leveraging decades of theoretical and empirical work rather than rediscovering patterns through engineering iteration. Both paths – inspiration and implementation – may enrich our understanding of how reasoning principles might transcend their original substrates.

References

Natasha Beeson, Edward JN Stupple, Malcolm B Schofield, and Paul Staples. Mental models or probabilistic reasoning or both: Reviewing the evidence for and implications of dual-strategy models of deductive reasoning. *Psychological Topics*, 28(1):21–35, 2019.

Felix J Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. *arXiv preprint arXiv:2410.13787*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael C Mozer, and Sanjeev Arora. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *Advances in Neural Information Processing Systems*, 37:19783–19812, 2024.

John H Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, 34(10):906, 1979.

Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. Meta reasoning for large language models. *arXiv preprint arXiv:2406.11698*, 2024.

Simeng Han, Tianyu Liu, Chuhan Li, Xuyuan Xiong, and Arman Cohan. Hybridmind: Meta selection of natural language and symbolic language for enhanced llm reasoning. *arXiv preprint arXiv:2409.19381*, 2024.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Ineke Imbo, Sandrine Duverne, and Patrick Lemaire. Working memory, strategy execution, and strategy selection in mental arithmetic. *Quarterly journal of experimental psychology*, 60 (9):1246–1264, 2007.

Li Ji-An, Hua-Dong Xiong, Robert C Wilson, Marcelo G Mattar, and Marcus K Benna. Language models are capable of metacognitive monitoring and control of their internal activations. *arXiv preprint arXiv:2505.13763*, 2025.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. Revise: Learning to refine at test-time via intrinsic self-verification. *arXiv preprint arXiv:2502.14565*, 2025.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, et al. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

Wenhao Liu, Xiaohua Wang, Muling Wu, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Aligning large language models with human preferences through representation engineering. *arXiv preprint arXiv:2312.15997*, 2023.

Tongxu Luo, Wenyu Du, Jiaxi Bi, Stephen Chung, Zhengyang Tang, Hao Yang, Min Zhang, and Benyou Wang. Learning from peers in reasoning models. *arXiv preprint arXiv:2505.07787*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594, 2023.

Thomas O Nelson and Louis Narens. Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation*, volume 26, pp. 125–173. Elsevier, 1990.

Nick Oh and Fernand Gobet. Monitor-generate-verify (mgv): Formalising metacognitive theory for language model reasoning. *Unpublished manuscript*, 2025.

Tian Qin, Core Francisco Park, Mujin Kwun, Aaron Walsman, Eran Malach, Nikhil Anand, Hidenori Tanaka, and David Alvarez-Melis. Decomposing elements of problem solving: What" math" does rl teach? *arXiv preprint arXiv:2505.22756*, 2025.

Lynne M. Reder. Strategy selection in question answering. *Cognitive Psychology*, 19(1):90–138, 1987.

Timothy C Rickard. Strategy execution in cognitive skill learning: an item-level test of candidate models. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30 (1):65, 2004.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*, 2024.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*, 2024.

Mark Steyvers and Megan AK Peters. Metacognition and uncertainty communication in humans and large language models. *arXiv preprint arXiv:2504.14045*, 2025.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

Yu Wang, Shiwan Zhao, Zhihu Wang, Heyuan Huang, Ming Fan, Yubo Zhang, Zhixing Wang, Haijun Wang, and Ting Liu. Strategic chain-of-thought: Guiding accurate reasoning in llms through strategy elicitation. *arXiv preprint arXiv:2409.03271*, 2024.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*, 2022.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. *arXiv preprint arXiv:2404.17140*, 2024.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. *Advances in Neural Information Processing Systems*, 37: 126032–126058, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A MGV (Monitor-Generate-Verify) Zero-shot Prompting Templates

We present the complete prompting templates used in our experiments. Variables are denoted in curly brackets {variable}.

A.1 Monitor Phase

```
Monitor
Problem: {problem}
{if t > 0:
evaluation_scores:
coherence: {score}
plausibility: {score}
consistency: {score}
goal_conduciveness: {score}}

Before you <think>, analyse this problem WITHOUT solving it.

Output format:
<monitor>
Task_Features: List 2-3 keywords describing the math concepts needed
Difficulty (0-1): Assess the challenge level
{if t > 0: Recalibrate your assessment based on the
evaluation_scores in previous_cycle. Lower scores
suggest higher actual difficulty than previously assessed.}
</monitor>
```

A.2 Generate Phase

```
Generate - Strategy Selection

Problem: {problem}
Task features: {task_features}
Difficulty: {difficulty:.3f}
Available strategies: {comma_separated_strategy_list}

Select ONE strategy from the list above that best fits this problem.

Output format:
Selected Strategy: [exact strategy name from list]
```

The strategies include: [multiplication_and_addition, basic_arithmetic, addition_and_multiplication, arithmetic_operations, multiplication, percentage_calculations, subtraction, algebra, subtraction_and_division, multiplication_and_division, multiplication_and_subtraction, addition_and_subtraction, percentage_calculation, addition_subtraction, average_calculation, subtraction_multiplication, division, addition, linear_equations, algebraic_reasoning].

```
Generate - Strategy Execution

Problem: {problem}
Strategy: {strategy_type}

Output format:
<think>
Work through the problem step by step using the {strategy_type} approach.
Show all calculations and reasoning.
{if t > 0:
previous_cycle:
difficulty: {difficulty}
task_features: {task_features}
strategy_used: {strategy_type}
reasoning: [full solution text]
evaluation_scores:
coherence: {score}
plausibility: {score}
consistency: {score}
goal_conduciveness: {score}
evaluation: [full evaluation text]}
</think>

<answer>
Output only the final numerical answer (no units or text).
</answer>
```

A.3 Verify Phase

```
Verify
Problem: {problem}
Solution: {solution}

Evaluate this solution on these dimensions (0-1 scale):

1. COHERENCE: Do the steps logically connect?
2. PLAUSIBILITY: Is the approach reasonable?
3. CONSISTENCY: Are calculations correct?
4. GOAL-CONDUCIVENESS: Does it answer the question?

Output format:
<evaluate>
Coherence: X.X
Plausibility: X.X
Consistency: X.X
Goal-conduciveness: X.X

Evaluation: [Provide a 2-3 sentence analysis explaining the scores.
Identify specific errors or strengths. For low scores, indicate
what went wrong (e.g., ”arithmetic error in step 3”, ”misunderstood
the question”, ”skipped crucial reasoning”). For high scores, note
what worked well. Be specific and actionable.]
</evaluate>
```

B Baseline Algorithm Implementations

Algorithm 2 SELF-REFINE algorithm (Madaan et al., 2023)

Require: input x, model M, prompts $\{p_{gen}, p_{fb}, p_{refine}\}$, stop condition $\text{stop}(\cdot)$

$y_0 = M(p_{gen} \| x)$ ▷ Initial generation (Eqn. 1)
for iteration $t \in 0, 1, \ldots$ **do**
  $fb_t = M(p_{fb} \| x \| y_t)$ ▷ Feedback (Eqn. 2)
  if $\text{stop}(fb_t, t)$ **then**
    break ▷ Stop condition
  else
    $y_{t+1} = M(p_{refine} \| x \| y_0 \| fb_0 \| \ldots \| y_t \| fb_t)$ ▷ Refine (Eqn. 4)
  end if
end for
return y_t

Algorithm 3 Self-Verification algorithm (Weng et al., 2022)

Require: input x, model M, prompt p_{cot}, candidates K, verifications P

$\{y_1, \ldots, y_K\} = M(p_{cot} \| x)$ with sampling ▷ Generate diverse answers
for each candidate y_k **do**
  $score_k = 0$
  $conclusion_k = \text{Rewrite}(x, y_k)$ ▷ Turn Q&A into statement
  for $p = 1$ to P **do** ▷ Verify multiple times
    for each condition c_i in x **do**
      $x_{masked} = \text{Replace}(x, c_i, \text{"X"}) + conclusion_k$
      $\hat{c}_i = M(p_{cot} \| x_{masked} \| \text{"Find X"})$ ▷ Backward reasoning
      if $\hat{c}_i = c_i$ **then**
        $score_k = score_k + 1$ ▷ Successful reconstruction
      end if
    end for
  end for
end for
return $\arg\max_k(score_k)$ ▷ Most verifiable answer wins