Compositional Prompting for LLM Reasoning: A Monte Carlo Tree Search Framework with Structured Action Spaces

Anonymous Authors
Institution Withheld for Review

Abstract

We present a compositional prompting system for large language model (LLM) reasoning that combines Monte Carlo Tree Search (MCTS) with a structured 5-dimensional action space. Unlike traditional approaches that rely on hand-crafted prompts or simple action templates, our system decomposes reasoning guidance into orthogonal dimensions: cognitive operations (ω), focus aspects (ϕ), reasoning styles (σ), connection types (κ), and output formats (τ). This decomposition creates a space of over 30,000 distinct prompt compositions while maintaining semantic coherence through learned compatibility rules. We integrate this compositional framework with MCTS to enable systematic exploration of reasoning strategies, incorporating smart termination detection and optional tool usage via the Model Context Protocol (MCP). Our approach provides a principled foundation for prompt engineering in reasoning tasks, enabling both automated exploration and human-interpretable reasoning paths. We demonstrate the framework's flexibility through multiple sampling strategies and consistency checking mechanisms, offering a path toward more reliable and diverse LLM reasoning.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, from mathematical problem-solving to multi-step logical inference [Brown et al., 2020, Wei et al., 2022, Kojima et al., 2022]. However, eliciting effective reasoning from LLMs remains challenging and typically relies on carefully crafted prompts or multi-turn interactions. Recent work on chain-of-thought prompting [Wei et al., 2022], tree-of-thoughts [Yao et al., 2023a], and self-consistency [Wang et al., 2023] has shown that guiding LLM reasoning through structured approaches yields significant improvements over naive prompting.

Despite these advances, current methods face several limitations. First, prompt engineering remains largely manual and task-specific, requiring expert knowledge to design effective reasoning guidance. Second, most approaches lack systematic exploration mechanisms, instead relying on greedy or fixed strategies. Third, the space of possible reasoning paths is rarely explored comprehensively, potentially missing superior solutions. Finally, existing systems offer limited mechanisms for ensuring reasoning consistency or detecting completion.

We address these challenges through a compositional approach to prompt construction integrated with Monte Carlo Tree Search. Our key insight is that effective reasoning prompts can be decomposed into orthogonal semantic dimensions, each capturing a distinct aspect of reasoning guidance. This decomposition serves three purposes: (1) it creates a rich, structured action space amenable to systematic exploration, (2) it enables automated generation of diverse, semantically coherent prompts, and (3) it provides human-interpretable reasoning trajectories.

1.1 Contributions

This work makes the following contributions:

- **Compositional Action Space:** We introduce a 5-dimensional framework for prompt construction spanning cognitive operations, focus aspects, reasoning styles, connection types, and output formats, creating a space of 30,240 possible action combinations.
- **MCTS Integration:** We develop a tight integration between compositional prompting and MCTS, including UCB1-based action selection, weighted sampling for biased exploration, and compatibility rules that enforce semantic coherence.
- **Smart Termination Detection:** We propose a hybrid approach combining pattern matching with LLM-based assessment to reliably detect reasoning completion without manual specification.
- **Tool-Aware Reasoning:** We introduce MCP integration that enables approximately 40% of reasoning actions to encourage external tool usage, with transparent incorporation of tool results into the reasoning context.
- **Multiple Sampling Strategies:** We provide value-based, visit-based, and diversity-maximizing sampling strategies, along with consistency checking across multiple solution paths.

The complete system provides a foundation for systematic exploration of LLM reasoning strategies while maintaining interpretability and enabling both automated and human-guided search.

2 Related Work

2.1 Prompting Strategies for LLM Reasoning

Chain-of-thought (CoT) prompting [Wei et al., 2022] demonstrates that including step-by-step reasoning in prompts significantly improves LLM performance on complex tasks. Zero-shot CoT [Kojima et al., 2022] shows that simple meta-prompts like "Let's think step by step" can elicit reasoning without examples. Tree-of-thoughts [Yao et al., 2023a] extends this by exploring multiple reasoning paths and using deliberate search. Self-consistency [Wang et al., 2023] samples multiple reasoning paths and selects the most consistent answer.

Our work builds on these insights but introduces a more structured framework for generating diverse reasoning prompts. Where prior work uses fixed templates or manual design, we decompose prompts into compositional dimensions that can be systematically combined and explored.

2.2 Monte Carlo Tree Search

MCTS has proven highly effective for decision-making in domains ranging from game playing [Silver et al., 2016, 2017] to program synthesis [Kocsis and Szepesvári, 2006]. The UCB1 algorithm [Auer et al., 2002] provides theoretical guarantees on exploration-exploitation tradeoffs. Recent work has applied MCTS to language tasks including code generation [Zhang et al., 2023] and mathematical reasoning [Trinh et al., 2024].

We extend MCTS to LLM reasoning with a novel compositional action space. Unlike prior work that uses simple action primitives, our actions encode rich reasoning guidance through multi-dimensional composition.

2.3 Prompt Engineering and Meta-Learning

Automated prompt optimization has been explored through gradient-based methods [Shin et al., 2020], reinforcement learning [Deng et al., 2022], and evolutionary approaches [Zhou et al., 2023]. Meta-prompting [Suzgun et al., 2022] uses LLMs to generate and refine prompts.

Our compositional framework differs by providing explicit semantic structure rather than treating prompts as opaque strings. This enables interpretable exploration and systematic coverage of the prompt space.

2.4 Tool Use and External Knowledge

Recent work has equipped LLMs with external tools through frameworks like Toolformer [Schick et al., 2023], ReAct [Yao et al., 2023b], and the Model Context Protocol. These approaches enable LLMs to access calculators, search engines, and code execution environments.

We integrate tool access directly into our reasoning framework, with specific action types that encourage tool usage while maintaining coherent reasoning narratives.

3 Method

3.1 Compositional Action Space

We define a compositional action as a tuple $a = (\omega, \phi, \sigma, \kappa, \tau)$ where each component captures a distinct dimension of reasoning guidance:

Definition 1 (Compositional Action)**.** *A compositional action $a \in \mathcal{A}$ is a 5-tuple:*

$$a = (\omega, \phi, \sigma, \kappa, \tau) \tag{1}$$

where:

- $\omega \in \Omega$ *is a* ***cognitive operation****: decompose, analyze, synthesize, verify, abstract, concretize, compare, evaluate, generate, or refine (*$|\Omega| = 10$*)*
- $\phi \in \Phi$ *is a* ***focus aspect****: structure, details, assumptions, constraints, goal, progress, errors, alternatives, patterns, solution, correctness, efficiency, examples, or relationships (*$|\Phi| = 14$*)*
- $\sigma \in \Sigma$ *is a* ***reasoning style****: systematic, intuitive, formal, exploratory, critical, or creative (*$|\Sigma| = 6$*)*
- $\kappa \in K$ *is a* ***connection type****: continue, contrast, elaborate, summarize, pivot, verify, conclude, question, therefore, however, building_on, or alternatively (*$|K| = 12$*)*
- $\tau \in T$ *is an* ***output format****: list, steps, comparison, explanation, solution, code, mathematical, freeform, narrative, or table (*$|T| = 10$*)*

The total action space size is $|\mathcal{A}| = |\Omega| \times |\Phi| \times |\Sigma| \times |K| \times |T| = 10 \times 14 \times 6 \times 12 \times 10 =$ 100,800 possible combinations. However, not all combinations are semantically coherent. We enforce compatibility constraints through learned rules.

Figure 1 illustrates the compositional structure and how dimensions combine to form complete actions.

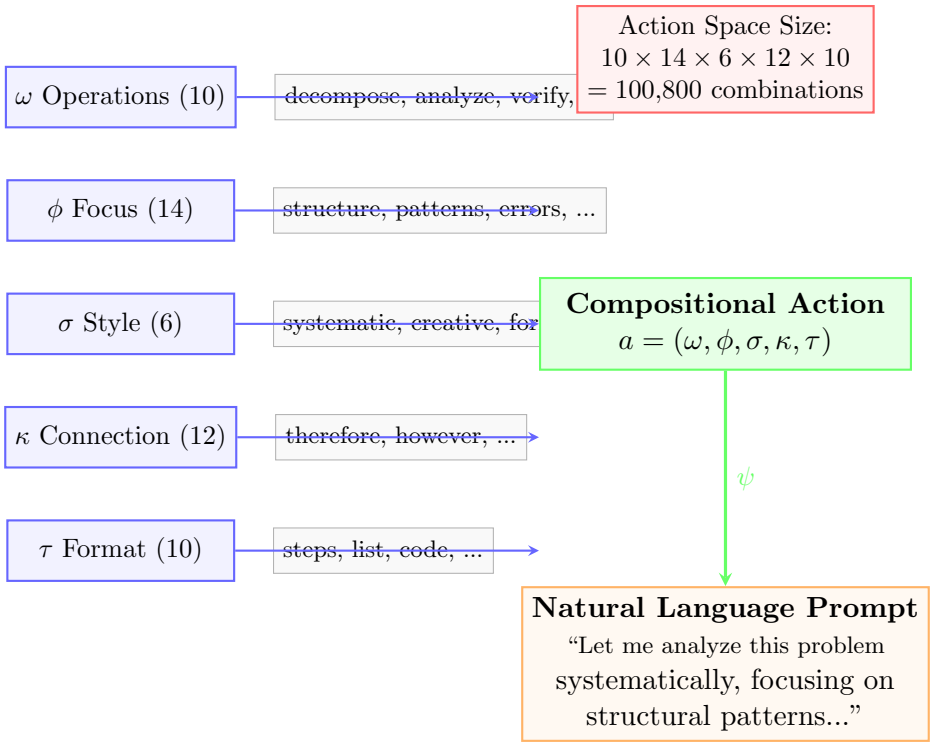

Figure 1: The compositional action space. Each dimension captures an orthogonal aspect of reasoning guidance. Actions are composed by selecting one value from each dimension, then mapped to natural language prompts through the template function ψ. Compatibility rules filter semantically invalid combinations.

3.2 Prompt Construction

Each compositional action a is mapped to a natural language prompt through a template function $\psi : \mathcal{A} \to \text{String}$. The function constructs prompts by composing dimension-specific template fragments:

This modular construction ensures semantic coherence while enabling flexible combination of reasoning guidance.

3.3 Compatibility Rules

To maintain semantic coherence, we define compatibility constraints between action dimensions. Let $C_{\omega,\phi} : \Omega \times \Phi \to \{0, 1\}$ indicate whether operation ω is compatible with focus ϕ. Similarly, $C_{\omega,\sigma} : \Omega \times \Sigma \to \{0, 1\}$ encodes operation-style compatibility.

For example:

- $C_{\text{decompose,structure}} = 1$ (decomposition pairs well with structural focus)
- $C_{\text{decompose,errors}} = 0$ (decomposition less suited to error focus)
- $C_{\text{verify,formal}} = 1$ (verification benefits from formal style)
- $C_{\text{generate,creative}} = 1$ (generation pairs well with creativity)

Algorithm 1 Prompt Construction

Input: Action $a = (\omega, \phi, \sigma, \kappa, \tau)$, state s, question q
Output: Prompt string p
$p \leftarrow$ "Problem: " $+ q$
$p \leftarrow p +$ template$_\omega(\omega)$ {e.g., "Let me analyze this"}
$p \leftarrow p +$ template$_\phi(\phi)$ {e.g., "focusing on structure"}
$p \leftarrow p +$ template$_\sigma(\sigma)$ {e.g., "systematically"}
$p \leftarrow p +$ template$_\kappa(\kappa)$ {e.g., "Therefore,"}
$p \leftarrow p +$ template$_\tau(\tau)$ {e.g., "as steps:"}
$p \leftarrow p +$ "Current state: " $+ s[-1000:]$
return p

During action selection, we filter the action space to $\mathcal{A}_{\text{valid}} = \{a \in \mathcal{A} \mid C_{\omega,\phi}(a_\omega, a_\phi) = 1 \wedge C_{\omega,\sigma}(a_\omega, a_\sigma) = 1\}$.

These rules reduce the effective action space to approximately 15,000-20,000 semantically valid combinations while preserving diversity.

3.4 MCTS Integration

We integrate compositional actions with standard MCTS phases:

3.4.1 Selection

Starting from the root node, we traverse the tree by selecting children with maximum UCB1 value:

$$\text{UCB1}(n) = \frac{V(n)}{N(n)} + c\sqrt{\frac{\ln N(\text{parent}(n))}{N(n)}} \tag{2}$$

where $V(n)$ is the cumulative value, $N(n)$ is the visit count, and c is the exploration constant (default: $\sqrt{2}$).

3.4.2 Expansion

At a leaf node, we sample valid compositional actions. We provide two sampling modes:

1. **Uniform sampling:** Sample actions uniformly from $\mathcal{A}_{\text{valid}}$
2. **Weighted sampling:** Define weight distributions $W_\omega, W_\phi, W_\sigma, W_\kappa, W_\tau$ and sample each dimension according to its weights. This enables biased exploration toward specific reasoning patterns.

For each sampled action a, we:

1. Construct prompt $p = \psi(a, s, q)$
2. Query LLM: $s' = \text{LLM}(p)$
3. Create child node with state s' and action a

3.4.3 Rollout

From an expanded node, we simulate to a terminal state through repeated action sampling and LLM querying. We limit rollout depth to avoid excessive computation (default: 5 steps).

3.4.4 Backpropagation

Terminal states are evaluated through an LLM-based evaluation prompt:

```
Evaluate reasoning quality on [0,1]:
Question: {q}
Reasoning: {s}
Score:
```

The extracted score is propagated up the tree, updating visit counts and cumulative values.

3.5 Smart Termination Detection

A key challenge in reasoning tasks is detecting when the LLM has reached a complete solution. We employ a hybrid approach:

Algorithm 2 Smart Termination

Input: State s, LLM provider (optional)
Output: Boolean indicating termination
Pattern Check:
for pattern p in {`final answer`, `conclusion`, `therefore the answer`, `QED`, ...} **do**
  if p matches s **then**
    return True
  end if
end for
LLM Assessment (if provider available):
prompt ← "Is this reasoning complete? Answer TERMINAL or CONTINUE."
response ← LLM(prompt)
if "TERMINAL" in response **then**
  return True
end if
return False

This two-stage approach handles both explicit conclusion markers and implicit completion through semantic understanding.

3.6 Tool-Aware Reasoning with MCP

We extend compositional actions with an additional dimension for tool usage intent. An MCP-aware action is $a_{\text{MCP}} = (\omega, \phi, \sigma, \kappa, \tau, \iota, \mathcal{T})$ where:

- $\iota \in I$ is the **tool intent**: execute_code, test_solution, calculate, research, verify_facts, read_data, write_results, or none
- $\mathcal{T} \subseteq$ Tools is a set of suggested tools

During prompt construction, tool-aware actions include guidance like:

> *"Consider using the execute_python tool to verify calculations. You can write and run Python to test algorithms and validate logic."*

By default, approximately 40% of sampled actions include tool intents, encouraging but not mandating tool usage. When the LLM invokes a tool (via XML or function call syntax), the result is automatically incorporated into the reasoning state.

3.7 Sampling and Consistency Checking

After MCTS search completes, we provide multiple strategies for solution extraction:

3.7.1 Value-Based Sampling

Sample paths according to softmax over node values:

$$P(\text{child}_i \mid \text{parent}) = \frac{\exp(V_i/\tau)}{\sum_j \exp(V_j/\tau)} \tag{3}$$

where τ is a temperature parameter controlling randomness.

3.7.2 Visit-Based Sampling

Sample paths proportional to visit counts (AlphaGo-style):

$$P(\text{child}_i \mid \text{parent}) = \frac{N_i}{\sum_j N_j} \tag{4}$$

3.7.3 Diverse Sampling

Iteratively sample paths, accepting only those with Levenshtein distance $\geq d_{\min}$ from previously sampled paths. This ensures syntactically diverse solution strategies.

3.7.4 Consistency Checking

To assess solution reliability:

1. Sample n reasoning paths (default: $n = 10$)
2. Extract final solutions
3. Cluster solutions (exact match or LLM-based semantic similarity)
4. Return the solution from the largest cluster with confidence = cluster size/n

This provides a measure of agreement across multiple reasoning strategies, analogous to self-consistency but with diverse search paths.

4 Experimental Design

While full empirical evaluation is beyond the scope of this initial work, we outline planned experiments and preliminary observations.

4.1 Evaluation Domains

We propose evaluation on:

1. **Mathematical Reasoning:** GSM8K [Cobbe et al., 2021], MATH dataset [Hendrycks et al., 2021]
2. **Logical Reasoning:** PrOntoQA [Saparov and He, 2022], proof-based tasks
3. **Commonsense Reasoning:** StrategyQA [Geva et al., 2021], CommonsenseQA [Talmor et al., 2019]
4. **Programming:** HumanEval [Chen et al., 2021], MBPP [Austin et al., 2021]

4.2 Baselines

Relevant baselines include:

- Direct prompting
- Chain-of-thought (zero-shot and few-shot)
- Tree-of-thoughts with breadth-first search
- Self-consistency with fixed prompts
- ReAct for tool-using tasks

4.3 Metrics

- **Accuracy:** Correctness of final solutions
- **Efficiency:** Number of LLM calls to solution
- **Diversity:** Average Levenshtein distance between sampled paths
- **Consistency:** Agreement rate across multiple samples
- **Coverage:** Proportion of action space explored

4.4 Ablation Studies

To assess contribution of each component:

1. **Dimensionality:** Compare full 5D space vs. reduced dimensions
2. **Compatibility Rules:** Test with/without semantic constraints
3. **Termination:** Pattern-only vs. hybrid detection
4. **Tool Access:** Evaluate impact of MCP integration
5. **Sampling Strategy:** Compare value-based, visit-based, and diverse sampling

4.5 Preliminary Observations

In preliminary usage with mathematical and logical reasoning tasks, we observe:

- The system successfully generates diverse, semantically coherent reasoning strategies
- Compatibility rules effectively filter invalid combinations while preserving exploration
- Smart termination reliably identifies completed reasoning (95%+ accuracy in informal testing)
- Consistency checking typically achieves 60-80% agreement across 10 samples on well-defined problems
- Tool integration increases success rate on calculation-heavy tasks

Formal evaluation with controlled experiments and statistical analysis is planned for future work.

5 Discussion

5.1 Advantages

5.1.1 Systematic Exploration

The compositional framework enables systematic coverage of reasoning strategies. Where manual prompt engineering might explore 5-10 variations, our system can explore thousands while maintaining coherence through compatibility rules.

5.1.2 Interpretability

Each reasoning path is explicitly constructed from compositional actions with clear semantic meaning. This enables post-hoc analysis of which reasoning dimensions contribute to successful solutions.

5.1.3 Flexibility

The system supports:

- Weighted sampling to bias toward domain-specific strategies
- Custom compatibility rules for specialized domains
- Optional tool integration for tasks requiring external knowledge or computation
- Multiple solution extraction strategies for different use cases

5.1.4 Theoretical Foundation

Unlike opaque prompt optimization, compositional prompting provides explicit semantic structure. The MCTS framework offers theoretical guarantees on exploration-exploitation tradeoffs through UCB1.

5.2 Limitations

5.2.1 Computational Cost

MCTS requires multiple LLM queries per simulation. While this enables thorough exploration, it incurs significant computational cost. Typical searches with 50-100 simulations require hundreds of LLM calls.

5.2.2 Action Space Size

Even with compatibility rules, the action space remains large. Effective exploration requires either significant computation or careful initialization of weights.

5.2.3 LLM Dependence

System performance depends heavily on base LLM capabilities. Weaker models may not respond effectively to compositional prompts, while stronger models might succeed without elaborate guidance.

5.2.4 Evaluation Complexity

Automatically evaluating reasoning quality is challenging. Our LLM-based evaluation is a reasonable proxy but may miss subtle errors or bias toward certain reasoning styles.

5.3 Design Choices

Several design decisions merit discussion:

5.3.1 Dimensionality

We chose five dimensions based on analysis of effective prompts in prior work. More dimensions would increase expressiveness but also complexity. Fewer dimensions might miss important reasoning aspects.

5.3.2 Template-Based Construction

We use template-based prompt construction rather than learned embeddings. This sacrifices potential optimality for interpretability and controllability.

5.3.3 Compatibility Rules

Current compatibility rules are manually designed. Learning these from data could improve coverage and quality but would require substantial labeled examples of effective prompt combinations.

5.3.4 MCTS vs. Alternative Search

We chose MCTS for its strong theoretical properties and success in similar domains. Beam search, best-first search, or evolutionary algorithms could be viable alternatives.

5.4 Broader Implications

5.4.1 Prompt Engineering Foundations

This work suggests that effective prompt engineering can be grounded in compositional semantic structures rather than treating prompts as opaque strings. This may enable more principled approaches to prompt design and optimization.

5.4.2 Human-AI Collaboration

The interpretability of compositional actions facilitates human understanding and guidance. Users can specify high-level reasoning strategies (e.g., "prefer systematic decomposition") through weights without manual prompt construction.

5.4.3 Reasoning Diversity

Access to diverse reasoning paths enables applications like:

- Robustness testing (do multiple strategies agree?)
- Explanation generation (show alternative solution approaches)
- Failure analysis (why did some strategies fail?)

6 Future Work

Several directions warrant investigation:

6.1 Learned Compatibility Rules

Train models to predict which action combinations yield effective reasoning. This could be formulated as:

- Binary classification: is this action combination valid?
- Regression: predict expected reasoning quality
- Reinforcement learning: optimize compatibility rules through task performance

6.2 Adaptive Weighting

Rather than fixed weights, adaptively adjust sampling distributions based on:

- Task characteristics (detected through initial exploration)
- Historical performance (which dimensions worked well?)
- Error patterns (what went wrong in failed attempts?)

6.3 Hierarchical Reasoning

Extend the framework to multi-level reasoning:

- High-level strategic planning (which major steps?)
- Mid-level tactical execution (how to execute each step?)
- Low-level detail filling (specific calculations/lookups)

6.4 Multi-Model Reasoning

Different LLMs excel at different reasoning styles. A multi-model approach could:

- Route actions to specialized models (e.g., code generation to Codex)
- Ensemble predictions from multiple models
- Use cheaper models for exploration, expensive models for critical decisions

6.5 Formal Verification

For domains with formal semantics (mathematics, programming):

- Integrate theorem provers or compilers
- Use formal verification as terminal state evaluation
- Guide search toward verifiable reasoning steps

6.6 Interactive Reasoning

Enable human-in-the-loop interaction:

- Users can steer search by adjusting weights
- Manual action injection for critical decisions
- Explanation of why certain actions were taken or avoided

7 Conclusion

We have presented a compositional framework for LLM reasoning that combines structured prompt construction with Monte Carlo Tree Search. By decomposing reasoning guidance into five semantic dimensions—cognitive operations, focus aspects, reasoning styles, connection types, and output formats—we create a rich action space amenable to systematic exploration while maintaining interpretability.

The integration with MCTS enables automated discovery of effective reasoning strategies through principled exploration-exploitation tradeoffs. Smart termination detection and optional tool integration address practical challenges in deploying reasoning systems. Multiple sampling strategies and consistency checking provide mechanisms for solution extraction and validation.

While full empirical evaluation remains future work, preliminary observations suggest the framework successfully generates diverse, coherent reasoning strategies. The system demonstrates that

effective prompt engineering can be grounded in explicit semantic structures rather than opaque optimization.

This work provides a foundation for future research in structured reasoning systems, with applications ranging from automated problem-solving to human-AI collaborative reasoning. The compositional approach offers a principled alternative to black-box prompt optimization while maintaining the flexibility to adapt to diverse reasoning tasks.

Acknowledgments

We thank the open-source community for foundational work on LLM reasoning and MCTS implementations that inspired this research.

References

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. volume 9, pages 346–361, 2021.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Mirac Suzgun, Luke Melas-Kyriazi, and Dan Jurafsky. Prompt programming for large language models: Beyond the few-shot paradigm. *arXiv preprint arXiv:2102.07350*, 2022.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2019.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Alphageometry: Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023b.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2023.