Solving Mystery Planning Problems Using Category Theory, Functors, and Large Language Models

Sumit Kumar Jha
Department of Computer Science
Florida International University
Miami, FL, USA
sumit.jha@fiu.edu

Susmit Jha
Computer Science Laboratory
SRI International
Menlo Park, CA, USA
susmit.jha@sri.com

Rickard Ewetz
ECE Department
University of Florida
Gainesville, FL, USA
rickard.ewetz@uf.edu

Alvaro Velasquez
Computer Science Department
University of Colorado Boulder
Boulder, CO, USA
alvaro.velasquez@colorado.edu

***Abstract*—Large language models (LLMs) have shown remarkable capabilities in natural language understanding and generation. However, they face challenges in solving complex planning problems, especially those obscured by altered terminologies and representations, known as mystery planning problems. This paper presents a novel approach leveraging category theory and functors to systematically map mystery planning problems to their canonical forms, enabling effective planning solutions. We demonstrate our methodology using the Mystery Blocks World domain, showcasing significant improvements in planning accuracy and efficiency. Contemporary LLMs, such as GPT-4 and Claude-3.5 Sonnet, conjecture the canonical form and the corresponding functor only by observing the structure of the mystery planning problem, and the resulting approach improves the accuracy of the LLM-based planning solution to 60% for problems with 4 blocks.**

***Index Terms*—Category Theory, Functors, Large Language Models, Planning, Autonomous Systems**

I. Introduction

In recent years, large language models (LLMs) such as GPT-4 and Claude-3.5 Sonnet and their variants have exhibited impressive capabilities in generating text, coding, answering mathematical queries and even predicting properties of biological sequences [1], [2]. Despite their strengths, these models encounter significant difficulties when tasked with solving complex planning problems, particularly those presented in obfuscated formats [3].

While formal verification of generated plans has enabled contemporary LLMs to synthesize plans with non-trivial accuracy [4], [5], this success does not hold true for Mystery domain planning as the accuracy of LLMs in these cases falls miserably even when supported by verification engines. This work introduces a category-theoretic framework that utilizes functors to translate mystery planning problems into their canonical forms, facilitating accurate and efficient problem-solving. Our main contributions in this paper are:

1) We introduce an approach leveraging functors to systematically map mystery planning problems to canonical forms corresponding to popular problems.
2) We demonstrate the application of this approach using the Mystery Blocks World domain, showcasing improvements in planning capability, pushing the accuracy to 60% for problems with 4 blocks.

Mystery Planning

```
(define (problem gen4)
  (:domain mystery-4ops)
  (:objects g j f b)
  (:init
    (harmony)
    (planet g)
    (planet j)
    (planet f)
    (planet b)
    (province g)
    (province j)
    (province f)
    (province b)
  )
  (:goal
    (and
      (craves g j)
      (craves j f)
      (craves f b)
    )
  )
)
```

Blocksworld Planning

```
(define (problem gen4)
  (:domain blocks-world)
  (:objects g j f b)
  (:init
    (ontable g)
    (ontable j)
    (ontable f)
    (ontable b)
    (clear g)
    (clear j)
    (clear f)
    (clear b)
    (armempty)
  )
  (:goal
    (and
      (on g j)
      (on j f)
      (on f b)
    )
  )
)
```

Fig. 1: Functor synthesized by a LLM maps the Mystery planning problem into the Blocks World problem.

3) We show that contemporary LLMs, such as GPT-4 and Claude-3.5 Sonnet, can conjecture the canonical form and the corresponding functor only by observing the structure of the mystery planning problem. For example, Fig. 1 shows a Mystery planning problem and its corresponding Blocks World problem synthesized automatically using the functor generated by GPT-4 in Table I of Sec. IV-B.

Our interest in solving Mystery planning stems from the desire to enable LLMs to operate in novel hithero unseen scenarios by mapping the problem to observed data and experiences using category theory and functors [6].

II. Related Work

A significant challenge in the deployment of LLMs is their propensity to produce hallucinations, which are outputs that are factually incorrect or nonsensical [7]. In the context of planning and synthesis [8], formal verification has been used to eliminate such hallucinations [4].

Recently, it has been argued that LLMs may enhance planning and decision-making tasks by leveraging their vast commonsense knowledge [9]. ADaPT [10], an approach that

decomposes tasks as needed, allowing LLMs to handle complex tasks by breaking them into simpler sub-tasks.

Category theory offers a powerful framework for understanding and formalizing various concepts in computer science and AI. Category theory [11], [12] has been used to structure and solve complex problems by defining relationships between different components systematically. To the best of our knowledge, we are the first to suggest the use of category theory and functors to deobfuscate mystery planning problems into other well-known planning problems.

III. Technical Approach

Our approach begins by providing the PDDL description of the mystery planning problem to an LLM. The LLM is then tasked with conjecturing a known planning domain from which this problem may have originated or to which this problem could be mapped. This involves the LLM identifying the category for both the original mystery domain and the conjectured planning domain. Based on these categories, the LLM is asked to construct a functor that translates objects and actions from one domain to the other.

Once the functor is formulated, it is used to translate specific problems from the mystery domain to the conjectured planning domain. Subsequently, we apply LLM-based planning using formal verification [4] to synthesize a verifiable plan within the conjectured planning domain. The functor is then employed again to map the synthesized plan back from the conjectured domain to the original mystery domain. The process is illustrated in Fig. 2.

It is acknowledged that the process may encounter failures at either the planning or conjecturing stages if the conjectured planning domain is not a suitable match. In such instances, feedback is provided to the LLM to facilitate the generation of a new planning problem or plan. While there is a possibility that the feedback loop may not yield successful results, our experiments have shown that contemporary models are capable of performing the necessary category-theoretic analysis and functor synthesis effectively.

A. Categories and Functors

We briefly recapitulate the definition of categories and functors and then discuss their use in planning problems.

Definition III.1. A category $\mathcal{C}$ consists of three components:

(i) A class of objects $\text{Ob}(\mathcal{C})$.

(ii) For each pair of objects $A, B \in \text{Ob}(\mathcal{C})$, a set of morphisms (or arrows) $\text{Hom}_{\mathcal{C}}(A, B)$ from A to B.

(iii) A binary operation called composition, defined on compatible pairs of morphisms. For any morphisms $u : A \to B$ and $v : B \to C$, there exists a morphism $v \circ u : A \to C$, read as "the composition of u followed by v".

These components must satisfy the following axioms:

(a) Associativity: For any morphisms $u : A \to B$, $v : B \to C$, and $w : C \to D$, the composition must be associative, i.e., $(w \circ v) \circ u = w \circ (v \circ u)$.

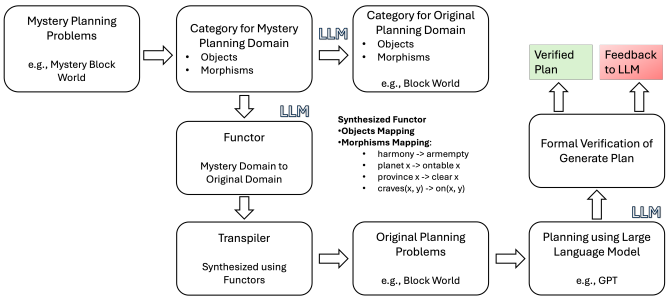

Fig. 2: Overview of our approach using category theory and functors to map mystery planning problems to canonical forms for LLM-assisted planning and formal verification.

(b) Identity: For each object $A \in \text{Ob}(\mathcal{C})$, there exists identity morphisms $\text{id}_A \in \text{Hom}_{\mathcal{C}}(A, A)$ and $\text{id}_B \in \text{Hom}_{\mathcal{C}}(B, B)$ such that for any morphism $u : A \to B$, we have $\text{id}_B \circ u = u$ and $u \circ \text{id}_A = u$.

In case of planning problems, the identity morphism simply denotes no changes in the state of the object. Composition of morphisms or actions in planning problems is indeed associative, as the grouping of the actions does not affect the state of the world. Hence, categories serve as effective mathematical models for planning problems.

Definition III.2. A functor F between two categories $\mathcal{C}$ and $\mathcal{D}$ is a structure-preserving mapping, consisting of:

(i) An assignment to each object $A \in \text{Ob}(\mathcal{C})$, an object $F(A) \in \text{Ob}(\mathcal{D})$.

(ii) An assignment to each morphism $u \in \text{Hom}_{\mathcal{C}}(A, B)$, a morphism $F(u) \in \text{Hom}_{\mathcal{D}}(F(A), F(B))$, such that the following conditions hold:

(a) Identity Preservation: For every object $A \in \text{Ob}(\mathcal{C})$, $F(\text{id}_A) = \text{id}_{F(A)}$.

(b) Composition Preservation: For all morphisms $u : A \to B$ and $v : B \to C$ in $\mathcal{C}$, $F(v \circ u) = F(v) \circ F(u)$.

We will observe that our synthesized functors generated by LLMs indeed satisfy identity preservation, since the identity function maps an entity to itself in both categories.

B. Mapping Mystery Planning Problems

The core idea of mapping Mystery planning problems to other well-known planning problems is to create a functor, say F, that can map objects in the category corresponding to the Mystery planning problem, say A, to an object, say $F(A)$, in the well-known planning problem. The functor F maps another object B in the Mystery planning problem to $F(B)$. Now, let f be a morphism from A to B. Then, the functor ensures that the morphism $F(f)$

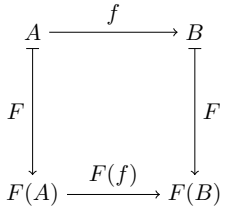

Fig. 3: A functor F mapping objects and morphisms from category $\mathcal{C}$ to category $\mathcal{D}$.

maps $F(A)$ to $F(B)$. In other words, the action of the functor on objects and morphisms is consistent, i.e., applying the morphism f on A and then the functor F on $B = f(A)$ has the same effect as first applying the functor F on A and then applying the morphism $F(f)$ on $F(A)$. Both of them produce $F(B)$. F maps objects and morphisms from the Mystery planning to the well-known planning domain.

By leveraging category theory, we can represent the Mystery planning problem and its canonical counterpart in a structured manner by asking the LLM to perform the following:

(i) Define the categories for Mystery planning problem and the known planning problem.
(ii) Specify the objects and morphisms in each category.
(iii) Formulate a functor to map between these categories, enabling translation of problems into canonical forms.

Prompt

Map the given Mystery Planning PDDL into a popular planning domain (referred to as domain X). Generate a planning PDDL in domain X that is analogous to the given Mystery Planning PDDL and compare the two. Define the categories for both the Mystery Planning problem and the generated problem in domain X. Specify the objects and morphisms in each category. Make sure that both PDDLs have the same number of predicates and actions. Formulate a functor that maps the Mystery Planning problem to the problem in domain X, illustrating how the objects and morphisms of the Mystery Planning problem correspond to those in domain X.

IV. Experiments

In this section, we describe the use of the GPT-4o large language model for mapping the Mystery planning problem into a known planning domain, specifically the classical Blocks World domain, using category theory. The LLM defines the categories, specifies the objects and morphisms in each category, and formulates a functor to map the Mystery Planning problem to the Blocksworld problem.

A. Categories, Objects, and Morphisms

To effectively map the Mystery planning problem to the Blocksworld domain, the LLM first defines the categories, objects, and morphisms for each problem.

1) Category for Mystery Planning ($\mathcal{C}_{mystery}$): The Mystery planning problem is defined by the following objects and morphisms. The objects include Provinces, Planets, Harmony, Pain, and Craves. The morphisms or actions that define the relationships between objects in the Mystery Planning domain are attack, succumb, overcome, and feast. These actions manipulate the state of the objects and are represented in the PDDL.

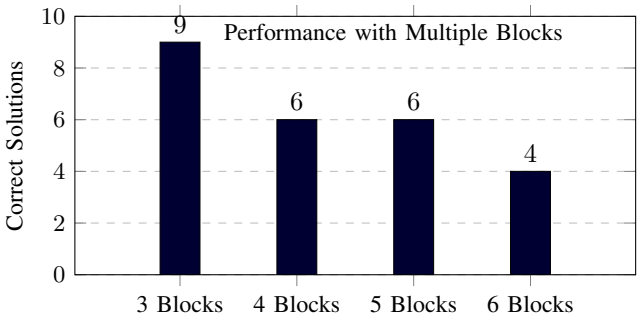

Fig. 4: Translating the Mystery Blocksworld problem using functor synthesized by GPT-4o leads to significant improvement in accuracy.

2) Category for Blocksworld Problem ($\mathcal{C}_{blocks}$): The Blocksworld domain is a well-known planning domain involving the manipulation of blocks. The LLM identifies this as the target planning PDDL for transforming the mystery PDDL into a well-understood planning problem. The LLM identifies the following objects:

(i) Blocks - The primary entities to be manipulated.
(ii) OnTable - The surface on which blocks can be placed.
(iii) Clear - A state indicating that a block is free,
(iv) On - A relation indicating a block is on top of another.
(v) Holding - A state indicating if an agent has a block.

The actions corresponding to the morphisms that define state transitions in the Blocks World domain are:

(i) pick-up - The action transitions a block from being on the table and clear to being held.
(ii) put-down - This action transitions a block from being held to being on the table and clear.
(iii) stack - This action transitions a held block to being on another clear block, changing the clear status of both.
(iv) unstack - This action transitions a stacked block to being held, restoring the clear status of the underlying block.

B. Functor Construction

Table I illustrates the functor F that maps the objects and morphisms from the Mystery planning problem ($\mathcal{C}_{mystery}$) to the Blocks World problem ($\mathcal{C}_{blocks}$). The LLM automatically generated the functor mapping the objects and the morphisms of the Mystery planning category to the Blocks World category. Using the functor F, we can transform a given Mystery planning problem into an equivalent problem in the Blocks World domain. An example of this transformation is in Fig. 1.

Mystery	**Blocks World**
Objects	
province	clear
planet	ontable
harmony	armempty
pain	holding
craves	on
Morphisms (Actions)	
attack	pick-up
succumb	put-down
overcome	stack
feast	unstack

TABLE I: LLM & Functor.

C. Mystery Blocks World Planning Results

The performance of GPT-4 on Mystery planning problem with multiple blocks after the category-theoretic mapping is shown in Fig. 4. The bar chart shows the number of correct solutions out of 10 problems for configurations with 3, 4, 5, and 6 blocks.

V. Generalization to Other Models

A. Claude Sonnet 3.5

We illustrate the functor F synthesized by the Claude Sonnet 3.5 model in Table II that maps the objects and morphisms from the Mystery planning problem to the Blocksworld problem. Comparing to Table I produced by GPT-4o, we notice that there are only two differences: (i) 'harmony' is mapped to 'balanced' instead of 'arm-empty', and '(ii) 'pain' is mapped to 'held' instead of 'holding'. These are minor idiomatic differences that do not change the semantics of the Blocks World problem.

Mystery	**Blocks World**
Objects	
province	clear
planet	ontable
harmony	balanced
pain	held
craves	on
Morphisms (Actions)	
attack	pick-up
succumb	put-down
overcome	stack
feast	unstack

TABLE II: Claude & Functor.

B. Llama 3.1 405B and Gemini 1.5 Pro

We queried the Llama 3.1 405B parameter model using the same prompt and obtained a mapping which was not correct. Llama's largest model had mapped both 'province' and 'planet' to 'block' and both 'pain' and 'craves to 'on', which was surprising. Smaller Llama 3.1 models also produced incorrect responses. We queried the Gemini 1.5 Pro model using the same prompt, and it produced an almost correct answer with only one deficiency: it swapped the assignments of 'province' and 'planet' to 'clear' and 'ontable' respectively.

VI. Conclusions, Limitations, and Future Work

In this paper, we have introduced an approach that leverages category theory and functors to systematically map mystery planning problems to their canonical forms, enabling more effective planning solutions with LLMs. Second, we demonstrated improvements in planning accuracy and efficiency in the Mystery Blocks World domain, with accuracy increasing to 60% for problems with 4 blocks. Third, we showed that contemporary LLMs can conjecture the canonical form and the corresponding functor by observing the structure of the mystery planning problem.

However, this work has known limitations. First, we conducted insufficient experiments across multiple planning problems to fully generalize the results. Second, we did not include discussions on our findings for smaller LLMs. Future work will address these limitations by providing rigorous proofs, expanding experiments to multiple planning problems, and including results from smaller LLMs. We will also extend our work to synthesis of verified code [6], [13]. Our interest in solving mystery planning stems from the desire to enable LLMs to operate in novel, hitherto unseen scenarios by using functors to transfer knowledge across domains.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) program, Reaching a New Energy Sciences Workforce (RENEW) under Award Number(s) DE-SC0024576 and DE-SC0024428. The authors acknowledge support from DARPA awards FA8750-23-2-0501 and HR00112490420. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the Department of Defense or the United States Government.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] M. Zvyagin, A. Brace, K. Hippe, Y. Deng, B. Zhang, C. O. Bohorquez, A. Clyde, B. Kale, D. Perez-Rivera, H. Ma, *et al.*, "Genslms: Genome-scale language models reveal sars-cov-2 evolutionary dynamics," *The International Journal of High Performance Computing Applications*, vol. 37, no. 6, pp. 683–705, 2023.

[3] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati, "On the planning abilities of large language models-a critical investigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 75993–76005, 2023.

[4] S. Jha, S. K. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, and S. Neema, "Dehallucinating large language models using formal methods guided iterative prompting," in *2023 IEEE International Conference on Assured Autonomy (ICAA)*, pp. 149–152, IEEE, 2023.

[5] S. K. Jha, S. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, R. Ewetz, and S. Neema, "Counterexample guided inductive synthesis using large language models and satisfiability solving," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, pp. 944–949, 2023.

[6] S. K. Jha, S. Jha, R. Ewetz, and A. Velasquez, "Co-synthesis of code and formal models using large language models and functors," in *Applications of Artificial Intelligence in Code Analysis (AICA) at the IEEE Military Communications Conference (MILCOM)*, 2024.

[7] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[8] S. K. Jha, *Towards automated system synthesis using sciduction*. University of California, Berkeley, 2011.

[9] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[10] A. Prasad, A. Koller, M. Hartmann, P. Clark, A. Sabharwal, M. Bansal, and T. Khot, "Adapt: As-needed decomposition and planning with language models," in *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252, 2024.

[11] H. Simmons, *An introduction to category theory*. Cambridge University Press, 2011.

[12] D. E. Rydeheard and R. M. Burstall, *Computational category theory*, vol. 152. Prentice Hall Englewood Cliffs, 1988.

[13] C. Spiess, D. Gros, K. S. Pai, M. Pradel, M. R. I. Rabin, S. Jha, P. Devanbu, and T. Ahmed, "Quality and trust in llm-generated code," *arXiv preprint arXiv:2402.02047*, 2024.