

Institute of Architecture of Application Systems

University of Stuttgart  
Universitätsstraße 38  
D-70569 Stuttgart

Masterarbeit

# **Leveraging LLMs for HTN Domain Model Generation via Prompt Engineering**

Vishesh Sinha

**Course of Study:** Computer Science

**Examiner:** Dr. Ilche Georgievski

**Supervisor:** Dr. Ilche Georgievski

**Commenced:** January 7, 2024

**Completed:** July 8, 2024



## Abstract

Domain models in Artificial Intelligence (AI) Planning represent the essential knowledge required for planning tasks in various sectors such as smart buildings, robotics, and autonomous vehicles. These models are critical for defining the actions, conditions, and effects that an AI planner uses to generate plans. However, due to their complex and hierarchical nature, creating these domain models is a complex and time-consuming process, especially for Hierarchical Task Network (HTN) based models.

Recognizing this challenge, this research investigates the use of Large Language Models (LLMs) to automatically generate domain models for AI planners. LLMs have previously demonstrated impressive abilities in generating code and other specifications from natural language inputs. While there has been preliminary research on using LLMs to produce non-hierarchical domain models specified in the Planning Domain Definition Language (PDDL), there has been no study focused on generating HTN domain models.

This thesis proposes a structured prompt engineering approach to evaluate the effectiveness of LLMs in generating domain models in Hierarchical Planning Definition Language (HPDL). We conducted experiments using several domain datasets, including cafeteria, blocks world, satellite, and a general domain, employing advanced models such as Generative Pre-Trained Transformer (GPT)-3.5, GPT-4, and GPT-4o. An SH planning system was utilized to verify the generated models.

The results indicate that LLMs can proficiently translate natural language into structured planning languages. However, challenges were encountered in generating plans that require arithmetic and physical reasoning. Despite these difficulties, the structured prompt technique significantly enhanced the potential of HPDL models, reducing the need for extensive human intervention. These findings suggest that LLMs can effectively bridge the gap between planners and human users, laying a foundation for future research in their use for creating advanced and accurate domain models. This research highlights the potential of LLMs as efficient tools for automated domain model generation in AI planning.



## **Acknowledgment**

I express deep gratitude to my supervisor, Dr. Ilche Georgievski, and the Institute of Architecture of Application Systems at the University of Stuttgart for providing me with the chance to undertake this thesis. The advice and expertise of Dr. Ilche Georgievski have greatly improved my knowledge and exploration of the subject since the beginning of the thesis. The thesis greatly benefited from his knowledge, patience, constructive feedback, and attention to detail.

I wish to convey my sincere gratitude to my family—my mother, father, and sister for their constant trust in me. Their constant support and confidence in my skills have consistently inspired me to pursue excellence.

I am grateful to my friends Pankhuri, Salman, Ria, Meghna, Deepanshu, and Faiz for their support and motivation. Their support has provided me with strength and motivation during the difficult periods of this journey.

Finally, I would like to express my gratitude to all those who have been involved in my thesis journey.



# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Research Context . . . . .	15
1.2	Research Problem . . . . .	16
1.3	Research Gaps . . . . .	16
1.4	Research Question . . . . .	16
1.5	Overview of Methodology . . . . .	17
1.6	Research Outline . . . . .	17
<b>2</b>	<b>Background Information</b>	<b>19</b>
2.1	AI Planning . . . . .	19
2.2	HTN Planning . . . . .	21
2.3	Challenges Associated with Manual Domain Model Creation . . . . .	22
2.4	HPDL . . . . .	23
2.5	Large Language Model (LLM) and Prompt Engineering . . . . .	24
<b>3</b>	<b>State of the Art</b>	<b>29</b>
<b>4</b>	<b>Methodology and Prompt Design</b>	<b>35</b>
4.1	Methodology Overview . . . . .	35
4.2	Prompt Design . . . . .	36
<b>5</b>	<b>Realisation</b>	<b>41</b>
5.1	Rationale for Selection of Domains, Planner, and Modeling Language . . . . .	41
5.2	Rationale for Selection of Tasks and Actions Across Domains . . . . .	43
5.3	Effective Prompt Techniques and Structures . . . . .	48
<b>6</b>	<b>Evaluation</b>	<b>51</b>
6.1	Evaluation Metrics . . . . .	51
6.2	Experimental Setup . . . . .	53
6.3	Model Quality Assessment . . . . .	55
<b>7</b>	<b>Conclusion and Outlook</b>	<b>63</b>
7.1	Summary of Findings . . . . .	63
7.2	Implications for Industry and Research . . . . .	63
7.3	Limitations . . . . .	64
7.4	Recommendations . . . . .	64
7.5	Future Work . . . . .	65
	<b>Bibliography</b>	<b>67</b>



## List of Tables

6.1	Complete evaluation metrics values for GPT-4o and GPT-4 in the blocks world domain for prompt structures S1, S2, and S3. . . . .	56
6.2	Complete evaluation metrics values for GPT-4o and GPT-4 in the satellite domain for prompt structures S1, S2, and S3. . . . .	58
6.3	Complete evaluation metrics values for GPT-4o and GPT-4 in the cafeteria domain for prompt structures S1, S2, and S3. . . . .	60
6.4	Comparison of total error value on GPT-4 and GPT-4o across different domain models for various prompt structures used to generate complete domain models. .	61



## List of Figures

2.1	Examples of Zero, One and Few Shot Prompting [BMR+20]	25
2.2	Prompt Chaining [GK24]	26
2.3	Chain-of-Thought Prompting [WWS+23b]	27
2.4	Generated Knowledge Prompting Method [LLL+22]	27
4.1	Prompt Structure	38



# Acronyms

**AI** Artificial Intelligence. 3

**API** Application Programming Interface. 30

**BERT** Bidirectional Encoder Representations from Transformers. 32

**BiLSTM-CRF** Bidirectional Long Short-Term Memory: Conditional Random Field. 32

**BNF** Backus-Naur Form. 43

**GPT** Generative Pre-Trained Transformer. 3

**HDDL** Hierarchical Domain Definition Language. 42

**HPDL** Hierarchical Planning Definition Language. 3

**HTN** Hierarchical Task Network. 3

**HVAC** Heating, Ventilation, and Air Conditioning. 20

**IPC** International Planning Competition. 29

**IRB** Institutional Review Board. 31

**LLM** Large Language Model. 7

**LLMs** Large Language Models. 3

**ML** Machine Learning. 32

**NASA** National Aeronautics and Space Administration. 41

**NLP** Natural Language Processing. 24

**PDDL** Planning Domain Definition Language. 3



# 1 Introduction

The thesis begins with this chapter that introduces the research topic and highlights the importance of using LLMs for AI planning, especially with HTN planning. It outlines the main research issue which focuses on whether it is feasible to use LLMs to automate domain model creation for HTN planning. The key aims of the study are then specified, including the creation of a logical LLM-based strategy for producing hierarchical domain models and assessing their usefulness. Finally, it provides an overview of the thesis structure, explaining how the following chapters are arranged.

## 1.1 Research Context

AI Planning is an essential element in diverse fields like intelligent buildings, robots, and self-driving vehicles. The process involves the automatic generation of sequences of actions to accomplish particular objectives while considering the limitations and changes in the surroundings [Geo23]. AI planning is the process of systematic generation of plans based on a defined set of rules and objectives. It allows systems to perform complex tasks autonomously, without requiring human intervention. This leads to enhanced efficiency and effectiveness in many applications [GB21].

HTN planning is a prominent technique in the field of AI planning widely utilised by researchers and practitioners [GA15]. HTN planning breaks down jobs into smaller, manageable sub-tasks, which makes it very efficient for practical applications including industrial automation, workflow management, and military operations. This hierarchical approach allows for systematic organization and execution of tasks, which is crucial for handling the complexity of real-world scenarios. The success of HTN planning in these applications underlines the importance of robust and accurate domain models.

Domain models are important inputs for AI planning algorithms as they specify the activities, objects, and relationships in the planning environment. These models have the foundational understanding of efficient planning which allows the algorithms to fully understand the surroundings and develop suitable strategies. However, manually developing these domain models, especially in a complicated syntax like the HPDL [Geo13], is a laborious and difficult task that requires a significant amount of time. This process requires significant expertise and attention to detail, making it vulnerable to errors and inefficiencies.

### 1.2 Research Problem

The manual creation of domain models presents significant challenges [MVV16]. It is a labour-intensive process that demands deep domain knowledge and precise execution. The complexity involved leads to errors and inconsistencies, which can undermine the effectiveness of the AI planning system. These issues highlight the need for more efficient and reliable methods to generate domain models. The traditional approach of manual model creation is time-consuming and also susceptible to human error, making it an unreliable method for developing robust AI planning systems.

### 1.3 Research Gaps

Due to the challenges associated with manual domain model creation, various techniques have been developed to automate this process. However, many of these techniques rely on plan traces as learning data, which are not always readily available [CMW13; Ji120; ZHY+09]. This dependency limits the applicability and effectiveness of existing methods. Additionally, LLMs have demonstrated remarkable capabilities in generating code and specifications from natural language input. This capability has extended to the generation of diverse specifications, such as software specifications [XYJ+23], graphs [CGP+23; MSF+23], robot task planning features [LPK+24a; LPK+24b] and process models [BSA23; KBK+23], however their application to AI planning has been limited. Initial research has explored the use of LLMs to generate non-hierarchical domain models specified in the PDDL [GVSK23; XYZ+23]. However, there is a noticeable research gap in leveraging LLMs for the automated generation of hierarchical domain models, which are crucial for HTN planning.

### 1.4 Research Question

Based on the identified gaps, this thesis addresses the following research question:

**To what degree is it possible to automatically construct domain models for HTN planning using LLMs?**

Addressing this question involves exploring the capabilities and limitations of LLMs in automating the creation of HTN domain models. HTN planning is an advanced approach used in AI to break down complex tasks into simpler, manageable subtasks, making it highly relevant for various industrial applications. The main aim of the research is to find out how well LLMs can help generate HTN domain models, which can improve planning systems in both industrial and research contexts.

## 1.5 Overview of Methodology

The thesis utilises the methodology of Prompt Engineering to answer this research question. Prompt engineering is a technique used to guide LLMs by designing specific prompts that generate desired outputs. This approach has been successfully applied in various fields, such as generating code, creating structured data, and developing software specifications. LLMs can be guided to generate domain models by providing detailed prompts that include examples and specific instructions.

This research develops a structured prompt framework designed to include essential sub-parts that guide the LLMs in generating accurate HPDL domain models. The process involves designing prompts that include necessary information about HPDL syntax, examples of domain models, domain knowledge, and step-by-step instructions. Experiments are conducted to assess the completeness, consistency, and correctness of the generated models and comparative analysis is used to understand whether the LLM-generated models can generate plans effectively compared to manually crafted models. Additionally, the adaptability and performance of LLMs-generated models across various domains and use cases are explored to identify strengths and limitations. Through this comprehensive approach, a robust framework for the automated generation of HPDL models using LLMs will be provided.

## 1.6 Research Outline

The content of this thesis is structured as follows:

**Chapter 2: Background Information** This section includes information about AI planning and its types, the challenges associated with generating manual domain model creation, HPDL, LLMs, and prompt engineering.

**Chapter 3: State of the Art** This chapter reviews existing literature on the use of LLMs in automated domain model generation, highlighting best practices and research gaps.

**Chapter 4: Methodology and Prompt Design** This chapter outlines the general steps of our methodology for developing a method to generate HTN domain models using LLM.

**Chapter 5: Realisation** This chapter gives an in-depth analysis of the reasoning and approach used to choose domains, planners, domain language, domain tasks, and domain actions in the study

**Chapter 6: Evaluation** This chapter describes the evaluation method, experimental setup, and assessment of all generated domain models.

**Chapter 7: Conclusion and Outlook** This chapter discusses the findings and their implications for industry and research and provides recommendations and future research directions.



## 2 Background Information

### 2.1 AI Planning

AI Planning is a crucial domain within AI that concentrates on automating the decision-making process. This specialised domain focuses on formulating strategies and sequences of actions to accomplish predetermined goals. The core concept of AI planning resides in its capacity to understand and control the elements of an environment to smoothly progress from an initial state to a desired goal state [Gef02].

AI planning is essential in AI domains as it includes the ideas of logical thinking, forecasting, and maximising efficiency. These concepts empower systems to carry out activities that traditionally require human intelligence. The planning process demands the identification of the present state of the environment, the definition of the desired end state, and the determination of the most efficient path or sequence of actions to bridge the gap between the two. This procedure is inherently complex and requires advanced algorithms capable of managing multiple variables and limitations.

The importance of AI planning cannot be overstated, as it forms the backbone of various practical applications ranging from autonomous robotics to intelligent resource management in healthcare. The ability to plan and execute actions autonomously allows systems to operate proactively, thereby significantly enhancing efficiency and effectiveness in various domains.

#### 2.1.1 Core Concepts and Methodologies in AI Planning

**States and Representations:** States in AI planning serve as representations of various configurations of the system. Every state represents a distinct momentary representation of all relevant variables and their conditions. These representations are essential as they serve as the fundamental basis for planning algorithms to function. Precise state representations allow the planning system to have a thorough understanding of the environment and make well-informed decisions.

**Actions and Transitions:** Actions are the basic procedures that move the system from one state to another. Every action is characterised by its preconditions, which are the necessary conditions for the action to be carried out, and its consequences, which are the alterations that take place in the state once the action is completed. The correlation between actions and states is crucial in the field of planning, as it determines the practicality and effectiveness of proposed plans.

**Goals and Objectives:** In the context of AI planning, goals refer to the specific outcomes that the planning system aims to achieve. Objectives establish a specific aim for the planning process and act as a standard for assessing the effectiveness of a plan. For instance, the objective of a robot will be to reach a predetermined destination.

**Plans and Strategies:** Plans are ordered sets of actions formulated to facilitate the transition of the system from its starting state to the desired end state. An effective plan guarantees that the necessary conditions for each activity are met and that the outcomes of these actions contribute incrementally towards achieving the desired objective. To create successful plans, advanced algorithms that can assess multiple possible sequences of actions and choose the most advantageous one are required.

### 2.1.2 Applications of AI Planning in Industry

AI planning is extensively used in various industries, greatly improving efficiency, decision-making, and automation. Here are a few examples of how AI planning is utilised in various industries:

#### Robotics

- **Autonomous Driving:** AI planning is essential for the functioning of autonomous vehicles, as it is employed to strategize secure and effective routes, navigate through changing traffic situations, and effectively handle unforeseen obstacles. This involves making immediate decisions to tackle difficult situations while guaranteeing the safety of passengers and complying with traffic regulations.
- **Manufacturing:** Robots use AI planning to optimise the order of processes during manufacturing. The objective of this optimisation is to improve accuracy and enhance overall efficiency. Planning algorithms can optimise the sequencing of jobs for robotic arms, resulting in reduced transition times and the prevention of conflicts.

#### Smart Buildings

- **Energy Management:** AI planning systems are utilised to optimize energy consumption in smart buildings. These systems can reduce energy usage and lower costs by controlling Heating, Ventilation, and Air Conditioning (HVAC) systems, lighting, and other utilities based on occupancy and weather forecasts. Planning algorithms adjust settings dynamically to maintain comfort while conserving resources.
- **Security Systems:** In the field of security, planning supports surveillance, access control, and emergency response plans. This enhances the overall security and safety of the building by ensuring that all systems work together seamlessly. For example, planning systems can be used to automate responses to security breaches, directing cameras, locking doors, and alerting authorities.

## Logistics and Supply Chain Management

- **Route Optimization:** AI planning is crucial for logistics organisations to optimise delivery routes. These technologies have the ability to greatly decrease operational expenses and environmental impact by minimising trip time and fuel consumption. Planning algorithms take into account variables such as traffic conditions, delivery timeframes, and the maximum load that vehicles can carry.
- **Inventory Management:** AI planning in supply chain management is the process of using AI to effectively coordinate supply networks and forecast demand. This results in ideal stock maintenance. It helps prevent excess inventory and stock shortages by using complex models to optimise the trade-off between inventory costs and service levels.

## 2.2 HTN Planning

HTN Planning is an advanced approach that decomposes tasks into smaller, manageable sub-tasks, creating a hierarchy of tasks and methods to achieve complex goals [GA15]. This approach is useful in dynamic and complex environments where flexibility and adaptability are crucial [AGPA22; GNLA17; GNN+17].

### 2.2.1 Key Concepts in HTN Planning

- **Tasks and Methods:** In HTN planning, tasks are high-level objectives that the planner aims to achieve. Methods provide the decomposition of these tasks into sub-tasks, outlining the steps required to accomplish each high-level task. This decomposition can be recursive to allow multi-level task hierarchies.
- **Operators:** Operators define the primitive actions that can be executed by the planner. Each operator specifies the preconditions that must be satisfied for the action to be performed and the effects that result from the action. Operators are the building blocks for constructing plans.
- **Task Networks:** Task networks represent the hierarchical structure of tasks and their relationships. They outline the dependencies between tasks and sub-tasks, ensuring that the planner follows a logical sequence in achieving the overall goal. Task networks facilitate the organization and execution of complex plans by breaking them down into manageable components.

### 2.2.2 Benefits of HTN Planning

- **Scalability:** HTN planning can handle large and complex planning problems by breaking them down into smaller, more manageable parts. This hierarchical decomposition allows the planner to focus on one sub-task at a time, making the planning process more tractable.

- **Flexibility:** The hierarchical structure of HTN planning aids dynamic adjustments and re-planning in response to changes in the environment. If a sub-task becomes infeasible, the planner can reconfigure the task network to find an alternative path to the goal.
- **Reusability:** Sub-tasks and methods developed for one planning problem can be reused in other problems with similar structures. This reuse enhances efficiency and reduces the effort required to develop new plans from scratch.

### 2.2.3 Industrial Applications of HTN Planning

- **Robotics:** HTN planning is used in autonomous robots to perform household tasks, industrial automation, and service robots in healthcare. For example, a household robot might decompose the task of cleaning a room into sub-tasks such as vacuuming, dusting, and organizing items [ZZL+10].
- **Military:** In military operations, HTN planning is applied to mission planning, where high-level strategic goals are decomposed into tactical and operational tasks. This allows for detailed planning of complex missions involving multiple units and resources [XJL+17].
- **Business Process Management:** HTN planning streamlines business operations by breaking down processes into sub-tasks. For instance, in order processing, the high-level task can be decomposed into sub-tasks like order verification, payment processing, and shipment scheduling, each with its own methods and operators [GFC13].

## 2.3 Challenges Associated with Manual Domain Model Creation

Creating a domain model manually for AI planning is a complex and time-consuming process that involves defining the states, actions, goals, and plans for a specific domain. McCluskey et al. [MVV16] state that the process of developing domain models for automated planning has numerous difficulties. A major obstacle is the procurement, validation, verification, and maintenance of planning domain models which requires an extensive and accurate understanding of the actions, processes, and events in the domain. This technique is laborious and requires great knowledge. Furthermore, the complexity is increased by the requirement for advanced representation languages and tools to facilitate and structure the building of domain models. Verifying and validating the accuracy, consistency, and completeness of domain models is essential but challenging. This task requires rigorous processes to confirm the truthfulness of statements and the practicality of proposed solutions. Another obstacle arises from the reliance on the initial encoding and the accuracy of the translation process, particularly when automated techniques are used to create domain models. In addition, the evaluation of the quality of domain models, including their functionality and suitability, continues to be an active field of study, requiring more reliable measures and approaches to assess and compare domain models effectively.

Challenges for HTN domain model generation were discussed by Georgievski and Aiello [GA15]. Creating domain models for HTN planners requires a substantial amount of domain knowledge, which can be a time-consuming and complex task. The relative effort needed to formulate domain knowledge depends on the capabilities and experience of the domain author. This includes their

understanding of the underlying planner and their expertise in the respective domain. HTN planners need advanced, knowledge-rich representations to function effectively. This can lead to improved performance but at the cost of increased effort in encoding the domain knowledge. The community has not yet found a technique to measure the effort required for encoding domain knowledge. The ambiguity in defining the required effort arises from the differences in capabilities and experience among domain authors. Another challenge is ensuring that the domain models are consistent and complete, and cover all possible situations that might be encountered during planning.

## 2.4 HPDL

HPDL is an advanced extension of the PDDL, specifically designed to support HTN planning. HPDL enables the specification of complex hierarchical planning problems by decomposing them into smaller, manageable sub-tasks, providing a structured and systematic approach to problem-solving [Geo13].

### Domain Definition

The domain definition in HPDL specifies the general structure of the planning problem. This includes defining the types of objects, predicates, functions, actions and tasks necessary for the domain.

- **Types:** Types define the various types of objects that exist within the domain.
- **Predicates:** Predicates are the logical statements that describe properties or relationships between objects.
- **Functions:** Functions are the numeric functions used to represent measurable quantities within the domain.
- **Actions:** Actions define the specific operations that can be performed within the domain. They include preconditions and effects
- **Tasks:** Tasks are the high-level goals that need to be decomposed into subtasks.

### Task Decomposition

Task decomposition is a critical aspect of HPDL, involving the specification of methods that break down high-level tasks into subtasks.

- **Methods:** These provide the decomposition of tasks, detailing how complex tasks can be divided into simpler, more manageable subtasks (an action or another task). Each method has its own set of preconditions.

### 2.4.1 Importance of HPDL in AI Planning

HPDL addresses the limitations of flat, non-hierarchical planning languages like PDDL by introducing a hierarchical structure that enhances the scalability, flexibility, and expressiveness of planning models.

- **Scalability and Manageability:** The hierarchical nature of HPDL allows planners to manage and solve large, complex problems by breaking them down into smaller, more manageable components. This decomposition facilitates more efficient problem-solving and reduces computational complexity.
- **Flexibility and Adaptability:** HPDL supports dynamic and flexible planning, enabling planners to adjust and re-plan as necessary. This is particularly important in real-world applications where environments are unpredictable and subject to change.
- **Reusability of Domain Knowledge:** Methods and operators defined in HPDL can be reused across different planning problems within the same domain. This reusability enhances efficiency and reduces the need for repetitive coding of similar planning tasks.
- **Improved Expressiveness:** HPDL allows for a richer representation of planning problems, capturing both high-level goals and the detailed steps needed to achieve them. This expressiveness is critical for accurately modeling complex domains and ensuring that all necessary subtasks are considered.

By leveraging the hierarchical structure and expressive power of HPDL, AI planners can tackle more complex and dynamic planning problems, leading to more robust and effective solutions across various domains.

## 2.5 LLM and Prompt Engineering

### 2.5.1 LLMs

LLMs represent a significant advancement in AI, designed specifically to understand and generate human language. These models, such as GPT-3.5, GPT-4, and GPT-4o, have been trained on extensive datasets comprising text from a wide array of sources. This comprehensive training enables LLMs to perform a wide range of language-related tasks such as answering questions, composing essays, translating languages, and generating computer code.

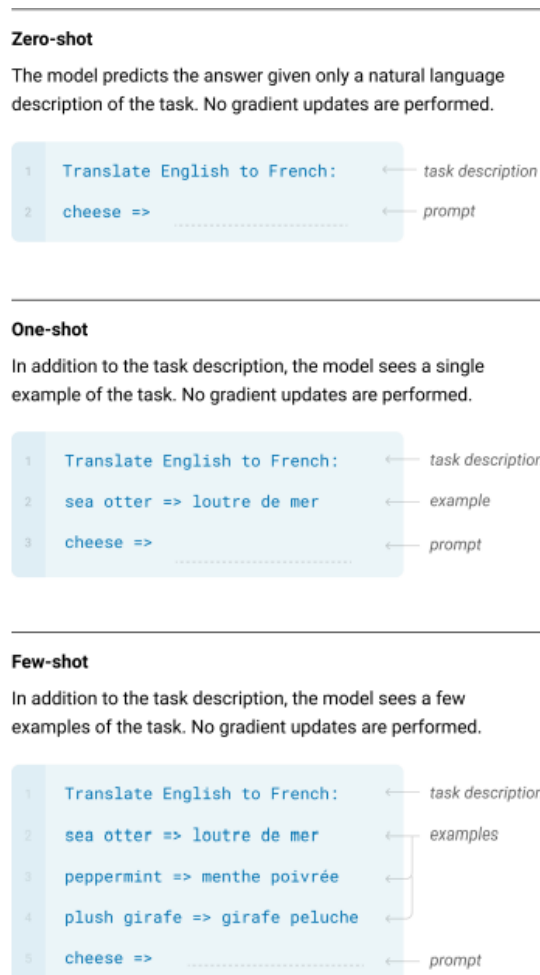
The impact of LLMs across various fields has been profound because of their advanced Natural Language Processing (NLP) capabilities. They can generate coherent and contextually appropriate text based on the prompts they receive [RWC+19]. However, it is crucial to understand how to effectively communicate tasks to these models to fully harness the potential of LLMs. This necessity leads us to the concept of prompt engineering.

## 2.5.2 Prompt Engineering

Prompt engineering is the practice of meticulously designing and structuring prompts to guide LLMs in generating the desired outputs. The quality of the prompt provided to an LLM significantly influences its effectiveness. Well-constructed prompts enhance the model's performance and increase the likelihood of producing accurate and relevant responses.

Prompt engineering encompasses various techniques aimed at optimizing interactions with LLMs. In the context of our research on generating HPDL domain models, several fundamental techniques in prompt engineering are employed to achieve optimal results.

- **Zero-shot and Few-shot Prompting:** Zero-shot prompting involves presenting the model with a task description without providing any examples and relying on the model's general knowledge to generate a response. In contrast, few-shot prompting includes a few examples to help the model better understand the task [BMR+20]. Figure 2.1 illustrates the examples of zero, one, and few-shot prompting methods.



**Figure 2.1:** Examples of Zero, One and Few Shot Prompting [BMR+20]

- **Prompt Chaining:** Prompt chaining involves combining multiple prompts in sequence to build up to a more complex task. Each prompt in the chain addresses a portion of the overall task, with the output of one prompt serving as the input or context for the next [WWS+23a]. Figure 2.2 shows a diagrammatic representation of the prompt chaining concept.

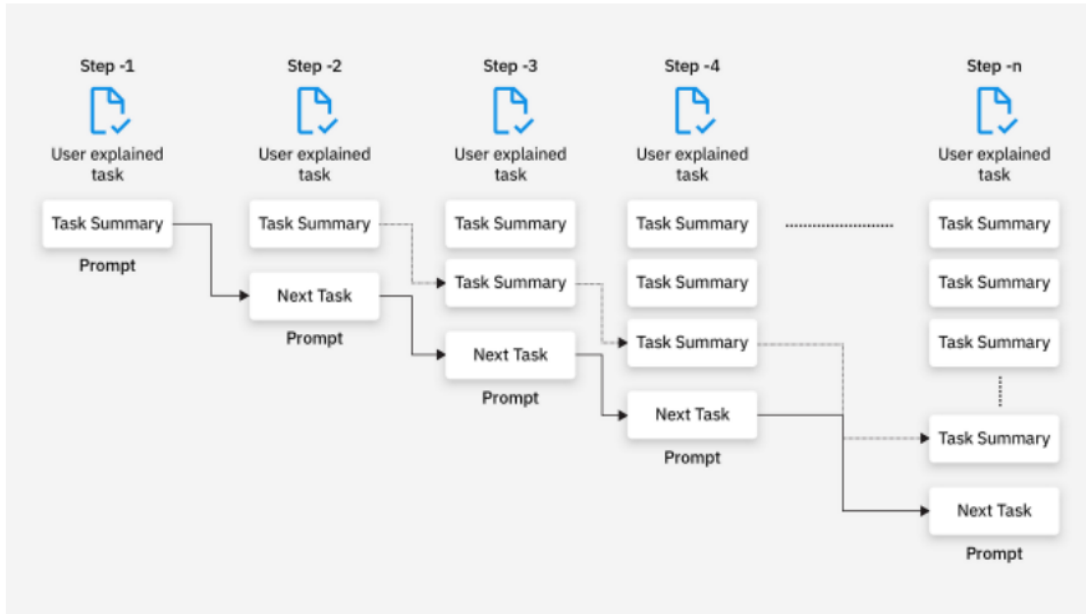


Figure 2.2: Prompt Chaining [GK24]

- **Chain-of-Thought Prompting:** Chain-of-thought prompting guides the model to explain its reasoning process step-by-step before delivering the final answer. This method helps the model follow a logical sequence, thereby improving the accuracy and coherence of its responses [WWS+23a]. Refer Figure 2.3 for understanding Chain-of-Thought concept.
- **Generated Knowledge Prompting:** Generated knowledge prompting involves instructing the model to generate intermediate pieces of knowledge to aid in arriving at the final answer. This technique assists the model in building a foundational understanding before tackling the main task [LLL+22]. Figure 2.4 shows the mechanism of the generated knowledge prompting method.

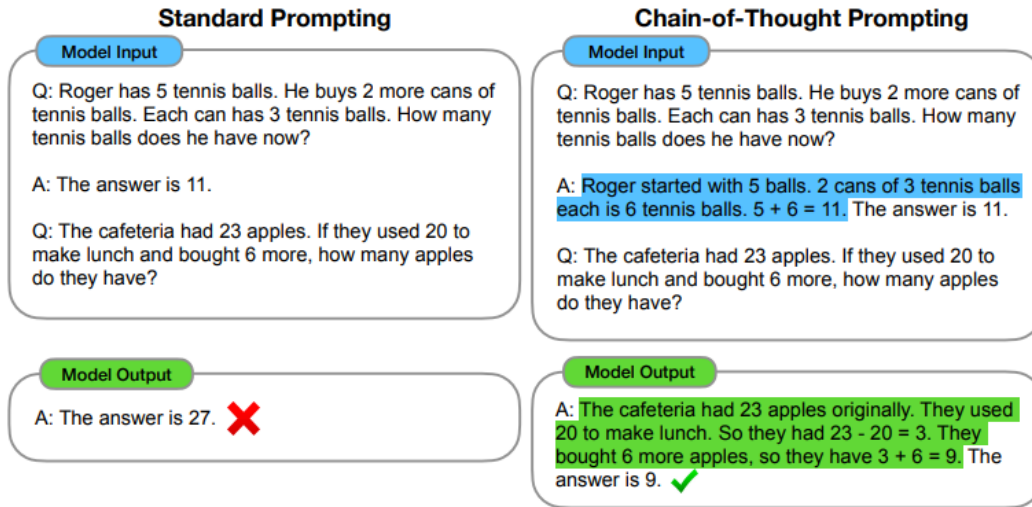


Figure 2.3: Chain-of-Thought Prompting [WWS+23b]

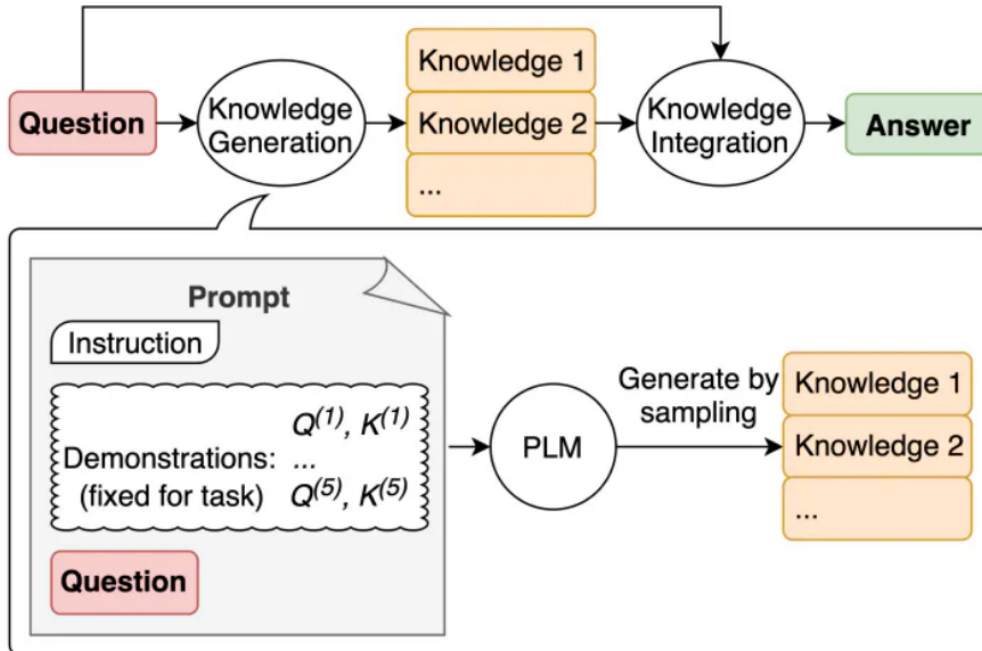


Figure 2.4: Generated Knowledge Prompting Method [LLL+22]



## 3 State of the Art

This chapter focuses on recent studies on the utilisation of LLMs to generate domain models. Additionally, it examines previous studies that have used LLMs to directly produce AI plans. The outcomes of these studies were largely unsuccessful, underscoring the necessity for further investigation into the utilisation of LLMs for the creation of domain models. Furthermore, there are research efforts dedicated to learning domain models from data. This chapter improves our understanding of the difficulties and strategies required to use AI for plan generation.

**“Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning”** Guan et al. [GVSK23] present a new method that extracts PDDL world models from LLMs and uses them with external domain-independent planners. This approach integrates the advantages of LLMs and conventional planners, guaranteeing the accuracy of the plan and including both explicit and implicit user limitations. The approach proposed consists of many crucial steps: LLMs use natural language descriptions to build PDDL action models, which are subsequently used by external planners to generate plans that fulfill certain goals. This approach addresses the constraints associated with directly utilising LLMs for planning, as they frequently encounter difficulties in generating accurate and implementable plans. The authors assessed the effectiveness of their methodology in two International Planning Competition (IPC) domains and a complicated household-robot domain, demonstrating superior performance compared to systems that exclusively depend on LLMs. One major benefit of this approach is the decreased need for human intervention, which enables users to proactively refine PDDL domain models by using an interactive approach that incorporates input from both PDDL validators and humans. Nevertheless, the approach relies on the assumption of complete observability and accurate determination of predicate values, which may not be applicable in real-world situations where perceptions are prone to noise. The complexity of the evaluation domains also presents difficulties. The experimental procedures employed encompass manual evaluation of generated PDDL models, utilisation of GPT-4 for model development, replaying dialogues for mistake correction, and translating user instructions into PDDL goal specifications. Conventional planners that are not specific to any one domain then conduct searches for plans using the PDDL models that have been created. This approach emphasises the importance of integrating symbolic and neural techniques to improve the dependability, adaptability, and efficiency of automated planning systems.

**“LLM+P: Empowering Large Language Models with Optimal Planning Proficiency”** Liu et al. [LJZ+23] combine LLMs with traditional planners to improve the planning abilities of LLMs. This technique aims to overcome the constraints of LLMs in terms of their ability to reason over long time horizons and provide practical plans. The methodology comprises three essential steps: transforming problem descriptions in natural language into PDDL files, solving these problems using classical planners, and transferring the solutions back into natural language. The study makes the assumptions that the robot can initiate LLM+P in response to user input, that a domain PDDL file outlining the robot’s capabilities is available, and that the context consists of a straightforward example problem and its PDDL file. The LLM+P architecture improves the planning

and problem-solving capabilities of LLMs without making any changes to the LLMs themselves. This is achieved by using external symbolic planners. An empirical evaluation was conducted to compare the performance of LLM-AS-P, which utilises LLMs directly for planning, and LLM+P on benchmark planning issues. The findings suggest that LLM-AS-P frequently fails to generate workable plans because it cannot consider preconditions. On the other hand, LLM+P consistently generates optimal plans for the majority of situations, with failures mostly attributed to difficulties in how the problem is defined. The inclusion of context, such as an example problem and its PDDL representation, is vital for the effectiveness of LLM+P. LLMs by themselves face difficulties in producing accurate PDDL problem files. The discussion highlights the framework's need for classical planners to enable LLMs, which requires specific assumptions such as the availability of a domain PDDL and a straightforward problem description. Promising areas for future research involve enhancing the capability of LLMs to automatically identify the appropriate circumstances and methods for using LLM+P, as well as decreasing the framework's reliance on information supplied by humans using fine-tuning. This study emphasises the possibility of integrating LLMs with traditional planning methods to address the limits of LLMs in planning complicated tasks. This opens up opportunities for the development of more sophisticated and capable planning systems.

**“Translating Natural Language to Planning Goals with Large-Language Models”** Xie et al. [XYJ+23] investigate the potential of GPT-3.5 to transform natural language instructions into PDDL goals. The study assesses the success of LLMs as intermediaries between human users and AI planners. The methodology uses GPT-3.5 variations to convert English instructions into PDDL objectives. The performance evaluation is conducted in the blocks world and ALFRED-L domains, consisting of tasks of different levels of complexity, including both fully and partially stated goals. The translation task is divided into three distinct steps: domain comprehension, goal deduction, and PDDL goal specification. The findings indicate that LLMs demonstrate strong performance in activities that have well-defined objectives, but face difficulties when the goals are only partially given. The performance of LLMs is highly influenced by the specific wording of stimuli, and they encounter challenges in arithmetic reasoning and making inferences about the physical environment. The study's findings indicate that although GPT-3.5 is capable of converting natural language instructions into PDDL goals, it still has difficulties with tasks that require numerical, spatial, and physical thinking. Therefore, additional research is required to address these obstacles. Research shortcomings that have been identified include the responsiveness of LLMs to prompts and the difficulty of conducting thorough prompt testing. The examination of subtasks is limited by the lack of Application Programming Interface (API) access, which hinders the ability to definitively validate success and identify specific transcribing problems. Although there are certain limits, this study helps us understand the possibilities and limitations of LLMs in converting natural language goals into structured planning languages. It also highlights the need for further research to improve their usefulness in complicated planning situations in the real world.

**“On the Planning Abilities of Large Language Models (A Critical Investigation with a Proposed Benchmark)”** Valmeekam et al. [VSM+23] conducted a thorough analysis of the planning abilities of LLMs and discovered that these models have a low success rate of approximately 3% at independently producing execution plans. The study assesses the performance of LLMs in three different scenarios: autonomous planning, heuristic guiding, and human-in-the-loop settings. The results indicate that LLMs are not effective in independently reasoning about actions and change. However, their performance shows a minor improvement with heuristic assistance from other planning agents or when humans are involved in the decision-making process. The methodology consists of three approaches: evaluating plans produced by LLMs directly (in autonomous mode),

---

using LLM-generated plans as input for automated planners to rectify (in heuristic mode), and presenting LLM-generated plans to human participants to observe their influence on problem-solving (in human-in-the-loop mode). Participants were assigned the challenge of solving a blocks world problem, with monetary rewards for well-devised plans. The results suggested that although LLMs may not operate well autonomously, they still provide valuable criteria that enhance the effectiveness of automated planners and assist human planners. The study presents strategies to improve logical thinking regarding the implementation of plans and expand performance evaluations to additional areas. The study establishes the objective of assessing the LLM planning capabilities and their efficacy in creating and verifying uncomplicated plans. It focuses on the importance of establishing a standardised benchmark for assessing the planning skills of LLM students. It also highlights the availability of tools and standards that may be accessed by the public for additional research purposes. The study employed a between-subjects randomised controlled user study, which was authorised by an Institutional Review Board (IRB). Participants were randomly assigned to either a control group or an experimental group. The experimental group received plan suggestions generated by LLM, while the control group did not. The study used 500 blocks world problem instances as stimuli. The objectives are to assess the LLMs' capacity to independently create and verify basic plans, as well as their efficacy in offering interpretative counsel. Potential areas for future research involve expanding benchmarks to encompass additional domains and improving the ability to reason about the implementation of plans. The dataset consists of the benchmark suite and automated evaluation tools created by the authors. The paper emphasises on the notable limitations of LLMs in autonomous planning while recognising their potential usefulness as predictive tools. The study highlights the necessity for additional research to enhance the planning capabilities of LLMs and broaden their usefulness in many sectors.

**“Generalized Planning in PDDL Domains with Pretrained Large Language Models”** Silver et al. [SDS+23] study the use of GPT-4 as a versatile planner to develop programs that effectively create plans for tasks in PDDL domains. GPT-4 uses techniques such as Chain-of-Thought summarization and automated debugging to increase performance by synthesising Python programs and creating plans. The study concludes that GPT-4 is an effective and versatile planner in PDDL domains, with the assistance of automated debugging. Remarkably, a mere two training challenges frequently prove to be enough for generalisation, demonstrating the model's capacity to adapt to novel problems within the same domain. Nevertheless, Chain-of-Thought summarisation exhibits a “non-uniform” effect, suggesting varying advantages. The study assesses the performance of GPT-4 in seven PDDL domains, by comparing it to four ablations and four baseline procedures. It evaluates the efficacy of Chain-of-Thought summarization and automated debugging in producing efficient plans and researches the viability of employing LLMs as universal planners. The results emphasise the strong and flexible nature of GPT-4, which requires only a small amount of training tasks. However, they also indicate the necessity for additional investigation into the varying effectiveness of Chain-of-Thought summarization and the dependence on automated debugging. Although the study shows positive outcomes, it acknowledges several constraints, such as the inconsistent effects of Chain-of-Thought summarization and the requirement for automated debugging. The absence of particular information regarding the PDDL domains and tasks employed during the evaluation may potentially restrict the applicability of the results. Overall, the study showcases the considerable capabilities of GPT-4 as a versatile planner in PDDL domains. The research highlights the significance of automated debugging and acknowledges the need for additional exploration to improve the reliability and coherence of the method.

**“A review of learning planning action models”** Arora et al. [AFP+18] in their survey offer a thorough examination of Machine Learning (ML) methods employed for acquiring planning action models, specifically focusing on the difficulties associated with encoding actions in complex real-world contexts. The review examines the development of automated planning from the 1960s onwards, highlighting the growing challenge of encoding actions as planning becomes more complex. This has resulted in the utilisation of ML to acquire action models. The paper systematically examines the current literature on ML techniques for planning action models, providing an in-depth review of the features of learning systems and the diverse techniques employed throughout the years. The techniques are carefully analysed, with a focus on their applicability and effectiveness in various contexts. The review additionally emphasises various unresolved matters, highlighting persistent difficulties and domains that require additional investigation. The objective of the paper is to provide a description of the characteristics of learning systems for planning action models, outline the learning techniques used, and highlight unresolved issues. The text explores how ML contributes to the development of action models in learning planning, specifically in complex real-world situations, by combining and enhancing existing knowledge. However, it does not present any new empirical findings. The survey highlights the importance of ongoing research to address the difficulties in automated planning and enhance the usefulness and effectiveness of planning systems in various fields.

**“Learning a Planning Domain Model From Natural Language Process Manuals”** Huo et al. [HTP+20] proposed a framework named FPTCP designed to automate the generation of planning domain models from NLP manuals. It specifically targets natural disaster management, such as typhoon contingency plans. FPTCP employs sophisticated NLP methodologies, such as Bidirectional Encoder Representations from Transformers (BERT) for word embeddings and Bidirectional Long Short-Term Memory: Conditional Random Field (BiLSTM-CRF) for part-of-speech tagging, to extract action ternaries, action representations, and pre/post-conditions. The purpose of this data is to create PDDL domain and problem files, which are solved to obtain the most efficient plans. This framework aims to automate the generation of planning models, specifically in the field of disaster management, thereby replacing time-consuming manual processes. The efficacy of the system was assessed using actual typhoon datasets, demonstrating its capacity to generate precise planning models based on natural language inputs. FPTCP is innovative in its automation of domain learning for PDDL models, specifically in the context of typhoon contingency planning. Potential enhancements involve minimising user involvement, facilitating knowledge dissemination to different areas, and extending the framework to encompass other planning models such as fully observable Markov decision processes and influence diagrams. Although FPTCP shows potential, it requires user involvement to improve its performance. The authors intend to reduce this interaction and broaden the framework’s scope.

**“A Comprehensive Framework for Learning Declarative Action Models”** Aineto et al. [AJO22] introduce a structured approach to assist researchers in developing systems for declarative action models, connecting them to AI tasks such as constraint satisfaction, optimisation, and belief propagation. Declarative action models are expressed as systematic searches that consider state constraints, offering a methodical way to comprehend state changes and make generalisations about objects. This methodology represents declarative action models as collections of state constraints and translates the learning task into a combinatorial search problem. The framework establishes a set of state constraints, referred to as an “alphabet” that govern the preconditions and effects of actions. The hypothesis space encompasses all models that can be created using this alphabet. Learning systems evaluate the suitability of potential models by comparing them to a collection

---

of learning examples. The paper highlights the necessity of a well-organized framework to tackle the difficulties of acquiring declarative action models in intricate AI applications. It specifically emphasises the importance of establishing a clear structure for the learning process, rather than focusing on empirical evaluation. The paper discusses the significance of acquiring declarative action models for deliberative tasks while acknowledging their frequent deficiency in terms of accuracy and comprehensiveness. It proposes the inclusion of reasoning under uncertainty as a way to address these limitations. The research gaps encompass the necessity for more comprehensive action models, difficulties in acquiring representative learning instances due to the rapid expansion of state space, managing incomplete observations, and creating scalable algorithms to accommodate various constraints and extensive datasets. Future research directions encompass augmenting the expressiveness of action models, refining the gathering and management of learning examples, and devising scalable algorithms for extensive and intricate datasets. This framework acts as a fundamental blueprint for progressing the field of learning declarative action models, by tackling existing constraints and delving into undiscovered opportunities.



## 4 Methodology and Prompt Design

### 4.1 Methodology Overview

This section outlines the general steps of our methodology for developing a method to generate HTN domain models using LLM. Each section is described in terms of its purpose and the criteria used for making selections.

#### 4.1.1 Selection of Modeling Language

The selection of a modeling language is a foundational element for representing planning problems and their solutions. The criteria for choosing the appropriate language include expressiveness, ensuring the capability to represent complex dependencies and hierarchical relationships between tasks. The language must support detailed and precise modeling of planning scenarios and accommodate hierarchical structures. Additionally, the chosen language should be well-documented and validated through academic or industry studies, providing a reliable basis for our work. Scalability is another key criterion that ensures the language can handle increasingly complex planning tasks as our framework evolves.

#### 4.1.2 Selection of Planner

The planner is responsible for generating and managing plans based on the models created using the selected modeling language via LLMs. The criteria for selecting the planner include compatibility, ensuring seamless integration with the chosen modeling language, and efficiency. This demonstrates the ability to handle complex models effectively. Scalability is essential, as the planner must be capable of managing large and complex planning tasks. Moreover, the planner's performance should be validated through empirical studies or benchmarks, providing evidence of its capability to meet our requirements.

#### 4.1.3 Selection of Domain Models

Selecting appropriate domain models is critical for evaluating the method for generating HTN domain models across a variety of scenarios. The domains should be relevant to real-world applications. This ensures the practicality and applicability of our approach. A range of tasks with different complexity levels should be used to thoroughly test the model's capabilities. Diversity in the selected domains ensures that the evaluation covers a broad spectrum of planning scenarios which is essential to assess the robustness and generalisability of the approach.

### 4.1.4 Selection of Language Model (LLM)

Choosing the appropriate language model is crucial for the successful generation of HTN domain models. The criteria for selecting the LLM includes language understanding and generation capabilities, ensuring the model can comprehend and produce complex task descriptions. The model should be pre-trained on diverse datasets to ensure robustness across various planning scenarios. Performance benchmarks, such as the ability to generate coherent and contextually accurate text, are essential for evaluating the LLM's suitability. Additionally, the model should be scalable, capable of handling large inputs and generating detailed outputs required for HTN domain models.

### 4.1.5 Criteria for Prompt Development

Developing criteria for prompt design is crucial for generating HTN domain models across various scenarios. The prompts should cover a wide range of planning scenarios. This will ensure a comprehensive evaluation of the model. The prompt should be designed to thoroughly assess the strengths and limitations of the method. This ensures that the LLM is not overfitted to a specific type of problem and can generalize well across different planning domains. Furthermore, the prompt should help identify specific areas where the model excels or needs improvement to guide future work.

## 4.2 Prompt Design

This section provides an overview of the design of prompts used to guide LLMs in generating HPDL domain models. It includes a detailed description of the structure of different prompt variations and their specific purposes, ensuring a thorough understanding of how prompts influence model generation.

### 4.2.1 Prompt Structure for LLMs

The approach proposes a structured prompt methodology to efficiently generate HPDL domain models using LLMs. This involves designing and testing a systematic prompt structure composed of several sub-parts [GVSK23; XYZ+23], which collectively contribute to the generation of the domain models. The key elements of the prompt structure are as follows:

#### **Components of the Prompt (P):**

**P1: Basic Information about HPDL** P1 provides foundational knowledge about HPDL, including its syntax and key concepts. This component is essential for ensuring that the LLM understands the basic framework and rules of HPDL. It includes definitions of important terms, the structure of HPDL, and an overview of its application in hierarchical planning. P1 establishes the foundation for the LLM to accurately understand and utilise the more specific information given in subsequent parts of the prompt. Refer Appendix A P1.

**P2: Sample HPDL Domain Model** P2 includes a sample HPDL domain model, actions, or tasks to provide the LLM with concrete examples. This component serves as a practical illustration of how HPDL is used in real scenarios. The sample model includes specific actions, their preconditions, and effects, which help the LLM understand how to construct similar models. By seeing a concrete example, the LLM can better grasp the application of HPDL syntax and concepts in practice, improving its ability to generate accurate and relevant domain models. Refer Appendix A P2.

**P3: Detailed Domain and Task Information** P3 gives detailed information about the specific action, task, or complete domain models that need to be generated. This is usually described in a combination of technical terms and natural language in a structured way that resembles the structure of the domain model. This component provides in-depth descriptions of the planning domain that include the specific tasks that have to be performed, the sequence of actions, and the dependencies between tasks. The technical descriptions ensure that the LLM has a precise understanding of the planning requirements and can generate models with domains and tasks accurately. P3 is crucial for understanding the detailed requirements of complex planning scenarios. Refer Appendix A P3.

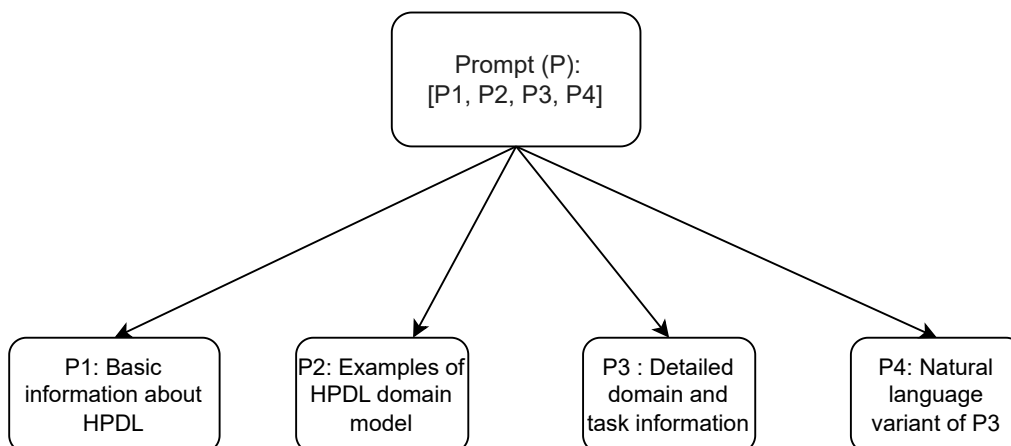
**P4: Natural Language Variant of P3** P4 is a variant of P3, in which the primary objective of the prompt is explained using natural language rather than technical terms. This component assesses the LLM's capacity to comprehend and convert natural language instructions into formal HPDL syntax. P4 evaluates the capacity of the LLM to process unstructured data and produce precise HPDL models by presenting task descriptions in natural language. This variation is especially valuable for assessing the adaptability and resilience of the LLM in comprehending and handling various forms of input. Refer Appendix A P4.

#### **Formulation of the Prompt (P):**

The complete prompt (P) is defined as a combination of its sub-parts:  $P=[P1, P2, P3, P4]$ . The generated domain model D is expressed as a function of P, where  $D=f(P)$ . Here, f represents the LLM's processing function of the structured prompt. This formulation highlights the importance of each sub-part in guiding the LLM to generate accurate and complete HPDL domain models. The combination of these components ensures that the LLM receives comprehensive guidance, from basic concepts to detailed task descriptions, enabling it to produce high-quality outputs.

### **4.2.2 Prompt Variations and Their Purposes**

Different combinations and permutations of the sub-parts are tested to understand their individual and collective impact on the efficiency and accuracy of the generated HPDL domain models. These variations are designed to identify the most effective prompt structures and to optimize the guidance provided to the LLM. By experimenting with different configurations, we can determine which components are most critical for generating accurate models and which can be omitted without significantly affecting performance.



**Figure 4.1:** Prompt Structure

**Complete Prompts:** Complete prompts involve using all sub-parts (P1, P2, P3) to provide maximum guidance to the LLM. By including detailed information about HPDL syntax (P1), a sample domain model (P2), and comprehensive task descriptions (P3), complete prompts ensure that the LLM has all the necessary information to generate accurate and complete domain models. This approach is useful for establishing a baseline performance level and understanding how much guidance is needed to achieve high-quality outputs.

**Natural Language Modifications:** In this variation, the detailed domain and task information (P3) is modified to be in natural language, forming (P4). This tests the LLM’s ability to interpret and translate natural language descriptions into formal HPDL syntax. It evaluates the LLM’s capacity to understand less structured input while still maintaining the accuracy of the generated models.

**Swapped Positions:** This variation involves swapping the positions of P2 and P3. We can assess the LLM’s flexibility in processing information presented in different sequences, by presenting the sample model (P2) after the detailed domain/ task information (P3). This helps to understand the impact of the order of information on the accuracy and completeness of the generated models.

**Zero-Shot Prompts:** Zero-shot prompts provide minimal instructions by omitting the sample HPDL domain model (P2) while still including basic HPDL information (P1) and detailed domain/ task information (P3). This tests the LLM’s capabilities to generate domain models with minimal input. Zero-shot prompts are valuable for assessing the LLM’s baseline performance and identifying areas where additional guidance can significantly improve model accuracy and completeness. This approach helps us understand the LLM’s strengths and limitations when operating with minimal input.

---

**Examples of Prompt: 4 different prompt structures used:****1. S1 : Combination of P1, P2, and P3:**

This variation includes basic HPDL information (P1), a sample model (P2), and detailed domain/ task information (P3). It provides a comprehensive prompt that balances foundational knowledge, practical examples, and detailed task descriptions.

**2. S2 : Modification of P3 to Natural Language (P4):**

This variation includes basic HPDL information (P1), a sample model (P2), and natural language task descriptions (P4). It tests the LLM's ability to interpret natural language input while also benefiting from the foundational and practical examples. Refer Appendix A S2

**3. S3 : Swapping Positions of P2 and P3:**

This variation changes the sequence by placing the sample model (P2) after the detailed domain/ task information (P3). It evaluates the LLM's flexibility in processing information presented in different sequences.

**4. S4: Zero-Shot Prompt :**

This prompt includes basic HPDL information (P1), and detailed domain/ task information (P3 ) in technical or natural language, but omits the sample model (P2). It evaluates the LLM's ability to generate domain models with minimal guidance.

By testing these variations, we aim to identify the most efficient and effective prompt structures for generating accurate HPDL models. The objective is to strike a balance between offering sufficient guidance to ensure high-quality output and minimising the amount of information required to attain such outputs. This balance is critical for developing a scalable and practical approach to using LLMs for HPDL model generation.

The design of prompts is a critical component in leveraging LLMs to automate the generation of hierarchical domain models in HPDL. By carefully structuring prompts and testing different variations, we can guide LLMs to produce complete, consistent and correct domain models. The detailed methodology for prompt design ensures that the proposed solution is robust and effective across various planning domains.



## 5 Realisation

This chapter gives an in-depth analysis of the reasoning and approach used to choose domains, planners, domain language, domain tasks, and domain actions in the study. This text outlines the factors to consider when choosing tasks and employing prompt approaches in a way that is efficient and successful. The purpose of this chapter is to provide a solid basis for the research by explaining the methodical strategy that was followed. This technique was designed to ensure that the findings are both reliable and relevant in the development of AI planning systems.

### 5.1 Rationale for Selection of Domains, Planner, and Modeling Language

#### 5.1.1 Selected Domains

The domains selected for this study were chosen based on their relevance, availability of data, and their potential to provide diverse contexts for evaluating the performance of hierarchical planning models. The selected domains include the satellite domain, cafeteria domain, blocks world domain, and the basic domain from the JSHOP2 distribution. Each of these domains offers unique characteristics and challenges, which are valuable for our research. All domain model encodings utilized in this study were based on the repository available in the PlanX Universe SH tool [Geo24]. This repository provided a standardized format for the domain models, ensuring consistency and reliability in the implementations.

**Basic Domain from JSHOP2 Distribution:** The basic domain model was based on the “basic” domain provided in the JSHOP2 distribution. This domain served as a control scenario to validate the performance of the generated models. It included standardized task descriptions and domain models essential for benchmarking the planning models.

The basic domain was selected to act as a baseline for comparing the performance of models across different domains. Its standardization offers standardized tasks and models which are crucial for consistent evaluation. The ease of implementation provided by this simplified domain gives quick validation and troubleshooting of generated domain models.

**Satellite Domain:** Data for the satellite domain was sourced from [AGA22; Geo24]. This domain models the National Aeronautics and Space Administration (NASA) space applications, focusing on missions that involve capturing images of spatial phenomena at various signal frequencies.

The satellite domain was chosen due to its relevance to real-world space exploration and its comprehensive hierarchical task structure, which includes energy management, instrument calibration, satellite orientation, and image capture. These detailed tasks provide a robust framework for testing LLM-generated domain models. The availability of well-defined models and extensive datasets allows for a thorough evaluation of model performance.

**Cafeteria Domain:** The data for the cafeteria domain was sourced from the work by Georgievski et al. [GNN+17] and Georgievski [Geo24], which explores planning and activity recognition for service coordination in intelligent buildings. This domain involves managing ceiling lamps in a smart building, focusing on tasks related to adjusting light settings and configurations. The scenarios and task descriptions provided by this domain were detailed and essential for the development and validation of our planning models.

The cafeteria domain was selected due to its relevance to modern technology and smart building management, which is a contemporary issue with practical applications in intelligent environments. The complex task management involved in this domain, including intricate tasks and scenarios, makes it ideal for testing the robustness of LLMs for generating domain models.

**Blocks World:** Data for the blocks world domain was sourced from [Geo24]. This domain revisits the classic blocks world problem, providing a foundational scenario for hierarchical task planning involving stacking and rearranging blocks.

The blocks world domain was chosen because it is a well-established benchmark in AI and planning research. The hierarchical tasks in blocks world are ideal for testing hierarchical planning models. The availability of detailed models and rich datasets gives an in-depth analysis of the models' performance. This domain serves as a benchmark for comparing new models against established standards, providing clear and structured tasks that are useful for initial model validation. Additionally, it helps in evaluating the effectiveness of models in a controlled setting, making it an essential component of our study.

Each domain contributes uniquely to our research, offering different contexts and challenges. These diverse domains highlight the strengths and areas for improvement across different types of tasks and settings, making them integral to our research.

### 5.1.2 Selected Modeling Language

The selection of a modeling language is a critical step in the design of our solution. We chose HPDL [Geo13] due to several reasons. Firstly, HPDL allows detailed representation of HTNs, making it well-suited for complex planning scenarios. This is particularly important for capturing the nuanced and multi-level planning required in many real-world applications. The expressiveness and structure of HPDL enable the representation of complex dependencies and hierarchical relationships between tasks, which is essential for accurate and effective planning.

Secondly, HPDL supports a more detailed syntax compared to PDDL, which is primarily used for flat, non-hierarchical planning. This flexibility allows for the definition of hierarchical relationships and task decompositions, making HPDL more suitable for capturing the detailed task hierarchies necessary for our research objectives. Thirdly, while the Hierarchical Domain Definition Language

(HDDL) also supports hierarchical planning [HBB+19], HPDL gives a more comprehensive set of features and robust syntax for defining detailed task hierarchies. The extended Backus-Naur Form (BNF) syntax of HPDL gives more flexibility and precision in modeling complex planning problems, which is not extensively supported in HDDL. This comprehensive syntax ensures that HPDL can handle the detailed and complex planning scenarios required for our solution.

Additionally, HPDL has extensive documentation and has been validated through numerous studies, providing a foundation for our experiments. This support ensures that our work is built on a well-established and thoroughly tested language, giving us confidence in the robustness and reliability of our solution. Lastly, HPDL is a scalable choice for future expansions of the planning framework due to its flexibility and comprehensive syntax.

### 5.1.3 Selected Planner

The SH planning system [Geo] was chosen as it has the capabilities to handle hierarchical structures and process HPDL models. It demonstrated excellent performance in empirical studies, particularly in processing complex hierarchical models. The SH planner has shown high performance and reliability, making it suitable for managing the extensive experimental framework required for our research. SIADEX is another planner that uses a version of HPDL [Ign22]. The version SIADEX employs includes some language constructs specific to SIADEX which is not generally used in HTN planning.

The SH planning system integrates with HPDL syntax and constructs, facilitating a smooth workflow from model generation to execution. This compatibility ensures that the planner can effectively process the detailed and complex task hierarchies represented in HPDL, enabling accurate and efficient planning. The SH planning system is capable of managing large and complex planning tasks which makes it suitable for our extensive experimental framework. The scalability of the SH planner ensures that it can handle the increasing complexity and the details required in future expansions of our planning framework.

## 5.2 Rationale for Selection of Tasks and Actions Across Domains

This section outlines the rationale behind the chosen tasks and actions, detailing their benefits and contributions to our research.

### 5.2.1 Blocks World Domain

The blocks world domain is a classic AI planning scenario that involves manipulating blocks to achieve specific configurations. This domain was chosen due to its well-established structure and complexity, which provide a robust framework for testing HPDL domain models. The tasks and actions selected for this domain are important for understanding AI's capability to handle ordered and structured activities. They also offer a variety of challenges that test the efficiency and accuracy of the hierarchical models generated by LLMs.

### Chosen Actions and Their Contributions

- **The action to pick up a block:** This enables the AI to interact with objects by picking up a block from the table. This action requires the block to be clear, on the table, and the agent's hand to be empty. Its effect ensures that the block is no longer on the table or clear, the agent's hand is no longer empty, and the agent is holding the block. This action is fundamental for initiating interactions with objects, setting the stage for more complex manipulations.
- **The action to put down a block:** This allows the AI to place a block onto the table. The precondition is that the agent is holding the block, and its effects ensure the block becomes clear, the agent's hand becomes empty, and the block is now on the table. This action is essential for completing tasks involving moving objects, demonstrating the AI's capability to release objects in designated places.
- **The action to stack a block:** This enables the AI to place one block on another. This action requires that the agent is holding the block and that the block being stacked on is clear. The effects of this action ensure the agent is no longer holding the block, the block being stacked on is no longer clear, the stacked block becomes clear, and the agent's hand becomes empty. Stacking blocks tests the hierarchical planning model's ability to create structures, requiring careful coordination and precise execution.
- **The action to unstack a block:** This action allows the AI to remove a block from another block. The precondition is that the block is on another block, the block being unstacked is clear, and the agent's hand is empty. Its effects ensure the agent is holding the block, the block being stacked on becomes clear, the unstacked block is no longer clear, and the agent's hand is no longer empty. Unstacking blocks is critical for tasks that involve deconstructing structures, and testing the model's ability to reverse previous actions accurately.

### Chosen Tasks and Their Contributions

- **The task to put a block on another block:** This is a compound task that involves placing one block on another. This task includes methods for both scenarios where the block is already on the other block (requiring no further action) and where the agent's hand is empty, requiring the AI to clear both blocks, put the second block on the table, and move the first block onto the second. This task evaluates the AI's ability to perform hierarchical planning steps involving multiple sub-tasks, demonstrating its ability to achieve composite goals.
- **The task to put a block on the table:** This task involves placing a block onto the table. This compound task includes methods for scenarios where the block is already clear and where it is not, requiring the AI to unstack and put down the block. This task validates the model's capability to perform essential operations such as unstacking and placing blocks, ensuring it can manage basic but crucial operations in a structured manner.
- **The task to move a block onto another block:** This task enables the AI to transfer a block onto another block. This task includes methods for scenarios where both blocks are clear and the agent's hand is empty, involving picking up and stacking the block, and where the block is not on the table, requiring unstacking and stacking. This task is critical for evaluating the AI's proficiency in executing a series of related actions that require both movement and placement, demonstrating the model's planning and execution capabilities.

- **The task to clear a block:** This task involves clearing a block. This compound task includes methods for scenarios where the block is already clear and where it is not, requiring the AI to clear and unstack the block above it. Clearing a block is a fundamental task that ensures subsequent operations can be performed without obstruction, testing the model's ability to recognize and address preconditions necessary for successful task execution.

### 5.2.2 Satellite Domain

The satellite domain was chosen due to its relevance to real-world space exploration and its comprehensive hierarchical task structure, which includes energy management, instrument calibration, satellite orientation, and image capture. This domain model has deep hierarchical as well as complex tasks. These detailed tasks provide a robust framework for testing LLM-generated domain models.

#### Detailed Actions and Their Contributions

- **Action to turn satellite:** This action allows the satellite to change its pointing direction from a previous one to a new one. It is effective only if the satellite is not already pointing in the desired direction, showcasing the AI's ability to efficiently redirect resources based on dynamic requirements. This action is essential for adjusting the satellite's orientation for various operations.
- **Action to switch on instrument:** The action to activate an instrument on the satellite checks for power availability before activation, highlighting effective power management. The outcome of turning the instrument on and marking it as uncalibrated prepares it for subsequent calibration tasks, underscoring sequential task management.
- **Action to switch off instrument:** This action makes power available again for other uses by deactivating an instrument. It requires that the instrument be currently powered on, reflecting the AI's capability to reverse operations as necessary to conserve resources or prepare for other tasks, thereby enhancing the satellite's adaptability in managing limited power supplies.
- **Action to calibrate instrument:** Calibration is pivotal for the accuracy of the instrument's functionality. This action emphasises AI's role in ensuring all preconditions are met before proceeding by relying on the instrument being powered and properly positioned. Successful calibration allows for high-quality operations such as accurate imaging, emphasizing the importance of precise setup.
- **Action to take image:** Capturing images in specified directions and modes is one of the satellite's primary functions. This action is dependent on the instrument being calibrated and correctly oriented, supported by the appropriate mode. It showcases the AI's ability to coordinate complex conditions to achieve mission-critical goals, highlighting its effectiveness in executing precise tasks that require the alignment of multiple system states.

### Chosen Tasks and Their Contributions

- **Task to turn satellite:** This task involves changing the satellite's orientation to a new direction, which is fundamental for various operations such as calibration and imaging. It tests the AI's ability to manage satellite orientation dynamically and appropriately based on mission needs, including scenarios where the satellite is already pointing in the required direction (requiring no action) or needs to be redirected.
- **Task to switch on instrument:** This task focuses on powering an instrument onboard the satellite, essential for enabling further tasks like calibration and image capture. It shows the AI's management of the satellite's power resources, covering scenarios where the instrument is already powered on and when it is not, ensuring efficient power usage.
- **Task to calibrate instrument:** Calibration is crucial for ensuring that instruments operate accurately. This task assesses the AI's proficiency in handling preconditions such as the satellite pointing in the right direction and the instrument being powered on. It incorporates methods for both scenarios where the instrument is already calibrated and where it isn't, demonstrating the AI's ability to perform necessary adjustments before main operations.
- **Task to capture image:** This task enables the satellite to take images in a specific mode and direction, based on the mission's requirements. It includes scenarios where the instrument is calibrated and pointing in the right direction, highlighting the satellite's operational capabilities in achieving the main objectives of the mission.
- **Task to manage power availability:** Managing power effectively is a critical aspect of satellite operations. This task involves making power available when it is consumed by operational instruments, either by switching off other instruments or managing power distribution effectively. It tests the AI's ability to prioritize and manage resources under constraints, ensuring continuous operational readiness.

### 5.2.3 Cafeteria Domain

The cafeteria domain was chosen due to its relevance in smart building technology and the complexities involved in managing lighting systems. This domain includes tasks related to turning on and off lamps, checking their status, and adjusting lighting levels in various areas, making it an ideal testbed for hierarchical planning models. The chosen tasks and actions simulate real-world scenarios of energy management and provide a framework to evaluate the AI's capability in handling dynamic environments.

### Chosen Actions and Their Contributions

- **Action to turn on a lamp:** This action enables the AI to control individual lighting units by turning on a specified lamp that is currently off. This action is crucial for managing lighting based on occupancy and environmental conditions, contributing to energy efficiency and optimal lighting control.

- **Action to turn off a lamp:** This action allows the AI to turn off a specified lamp that is currently on. This action is essential for reducing energy consumption when an area is unoccupied or adequately lit by natural light, demonstrating the AI's ability to manage energy effectively.
- **Action to mark a lamp as checked:** This action enables the AI to log the status of a lamp in a specific area, ensuring that each lighting unit is accounted for during the planning process. This action supports systematic verification and maintenance of the lighting system.
- **Action to mark an area as checked:** This action helps the AI ensure that an entire area has been evaluated for lighting needs. This contributes to comprehensive area management, ensuring that all sections are adequately monitored and controlled.
- **Action to uncheck all lamps in an area:** This action allows the AI to reset the checked status of all lamps within a specific area, preparing the system for a new evaluation cycle. This is important for maintaining up-to-date status information in dynamic environments.
- **Actions to increase or decrease light levels, either directly or due to nearby sources:** This action enables the AI to adjust lighting based on real-time conditions. These actions ensure that lighting levels are optimized for comfort and energy efficiency, directly impacting the environmental control within the cafeteria.

### Chosen Tasks and Their Contributions

- **Task to switch on all lamps in an area:** This task ensures that the AI can systematically illuminate a specified section of the cafeteria. This compound task includes methods to handle scenarios where lamps are either off or already on, ensuring that lighting levels are increased appropriately. This task demonstrates the AI's ability to manage multiple units cohesively.
- **Task to turn off all lamps in an area:** This task allows the AI to systematically reduce lighting in unoccupied or sufficiently lit areas. This compound task includes methods to handle different conditions, ensuring that energy consumption is minimized while maintaining necessary lighting. This task is critical for evaluating the AI's capability in managing resource efficiency.
- **Tasks to adjust light levels by increasing or decreasing for specific lamps:** This task ensures that the AI can fine-tune lighting based on real-time data. This compound task includes methods to handle different lighting requirements, ensuring that each area maintains optimal light levels. These tasks highlight the AI's adaptability and precision in environmental control.
- **Tasks to adjust lighting in a room based on presence or absence:** This task enables the AI to dynamically manage lighting for entire rooms, adjusting levels based on occupancy and light conditions. This task showcases the AI's ability to handle complex, multi-faceted environments and ensure efficient resource usage.

### 5.2.4 Basic Domain

The basic domain was selected as a control scenario to validate the performance of the generated models. This domain includes simple tasks such as picking up and dropping objects, providing a straightforward environment to test the core functionalities of the hierarchical planning models. The simplicity of this domain makes it ideal for benchmarking and ensuring that the AI can perform fundamental operations correctly before being applied to more complex scenarios.

#### Chosen Actions and Their Contributions

- **Action to pick up:** This action allows the AI to grasp a specified object. This action has no preconditions, making it a basic and essential function for object manipulation. Its primary contribution is to test the AI's ability to initiate interaction with objects, forming the foundation for more complex tasks.
- **Action to drop:** This action enables the AI to release a specified object, provided the agent has the object. This action is crucial for completing tasks that involve moving or rearranging objects, demonstrating the AI's capability to manage possession and release accurately.

#### Chosen Tasks and Their Contributions

- **Task to swap two things:** This task makes AI exchange two objects. This task includes methods for scenarios where the AI either has the first object or the second object, requiring it to drop the current object and pick up the other. This task is essential for evaluating the AI's ability to perform basic and structured operations that involve multiple steps and conditions. It ensures that the AI can handle sequences of actions effectively.

## 5.3 Effective Prompt Techniques and Structures

Through experimentation with different prompt structures and strategies, several best practices were identified. One important best practice is ensuring clarity and detail in prompts. Prompts that are clear and detailed significantly enhance the model's understanding of task requirements. This involves specifying the HPDL model structure, defining each component, and outlining the relationships between actions and tasks. Such detailed prompts provide the model with a comprehensive guide to follow, resulting in more accurate and coherent outputs.

Another strategy is the use of sequential instruction. Breaking down the task into smaller, manageable steps allows the model to understand the sequence in which actions and tasks should be generated. Incremental information delivery ensures that the model can focus on one aspect of the task at a time, maintaining logical consistency and reducing errors, as each step builds on the previous one. This step-by-step approach is beneficial in complex hierarchical planning tasks.

Incorporating contextual examples within the prompt is also highly beneficial while generating plans. Relevant examples help the model understand the expected format and structure of the output. Examples serve as a reference in guiding the model to produce similar outputs.

Prompt chaining combines multiple prompts in a sequence that helps in the gradual build-up of complex tasks. Each prompt introduces a new element, which is then integrated into the overall model. This approach ensures that the model has a thorough understanding of each component before advancing to more complex elements. Prompt chaining is especially useful in complex tasks that require an understanding of various interconnected components.

### 5.3.1 Prompting Techniques Utilized in Our Study

1. **Zero-shot and Few-shot Prompting:** Zero-shot prompting requires instructing the model to perform a task without providing any examples and relying solely on the given instructions. In contrast, few-shot prompting includes a few examples to aid the model in comprehending the task better.

In our research, we utilised few-shot prompting by incorporating a detailed example from the blocks world domain. This example provided a clear template for the model, demonstrating how to structure the HPDL domain model for similar tasks. By offering a well-defined example, the model was able to generate outputs that were correctly formatted and logically structured. This approach was particularly beneficial for tasks that required specific sequences of actions and conditions, as it established a clear precedent for the model to follow.

2. **Prompt Chaining:** Prompt chaining involves using multiple prompts in sequence to build up to a more complex task. Each prompt builds on the information provided in the previous one, guiding the model step-by-step.

Our prompt structure implemented chaining by first explaining the general guidelines, then breaking down the HPDL components, followed by providing an example and finally detailing the domain and tasks to be performed. This sequential approach ensured that the model had a clear understanding of each component before progressing to the next. This method was particularly effective in maintaining logical consistency and ensuring that all necessary preconditions and effects were accurately represented, ultimately leading to a coherent final domain model.

3. **Chain-of-Thought Prompting:** Chain-of-thought prompting guides the model to explain its reasoning process step-by-step before delivering the final answer. This method aids the model in following a logical sequence in its output generation.

Although we did not explicitly require the model to explain its reasoning, the detailed guidelines provided effectively served this purpose. By breaking down the task into logical steps and components, the prompt implicitly guided the model through a chain-of-thought process. This structured approach helped the model follow a logical sequence in generating the domain model, ensuring that each part of the model was built on a solid understanding of the preceding steps.

4. **Generated Knowledge Prompting:** Generated knowledge prompting involves asking the model to produce intermediate pieces of knowledge that contribute to the final answer. This method helps the model build a comprehensive understanding of the task.

In our approach, the breakdown of domain components and detailed instructions acted as intermediate knowledge parts. By prompting the model to generate specific parts of the HPDL domain model step-by-step, we ensured that each component was accurately constructed. This method allowed the model to integrate these intermediate parts into a combined final output, demonstrating a clear understanding of the domain's overall structure and requirements.

These strategies and insights will guide future research and development and help in enhancing the capabilities and applications of HPDL models. Our approach demonstrates the potential of LLMs to effectively generate complex planning models, providing valuable tools for the advancement of AI planning systems.

# 6 Evaluation

## 6.1 Evaluation Metrics

This section outlines a framework for evaluating HPDL domain models generated by LLMs. The evaluation metrics are divided into three main categories: General Evaluation [GVSK23], Hierarchy Evaluation, and Basic Syntax Error Evaluation. These metrics aim to thoroughly assess the generated models, identify discrepancies, and guide improvements.

### 6.1.1 General Evaluation

**Count of General Issues:** This includes:

1. **Removal of irrelevant preconditions in actions:** Identifying and eliminating unnecessary preconditions from actions.
2. **Addition of missing precondition in actions:** Ensuring all necessary preconditions are present in actions.
3. **Replacement of incorrect predicates:** Correcting wrongly defined or used predicates anywhere in the domain model.
4. **Removal of irrelevant effects in actions:** Removing effects that do not contribute to the intended outcome.
5. **Addition of missing effects in actions:** Adding essential effects for correct action execution.
6. **Inclusion of missing parameters in actions or tasks:** Ensuring all required parameters are included.
7. **Removal of irrelevant parameters in actions or tasks:** Ensuring all required parameters are included.

This metric measures the approximate distance between a generated action in HPDL model and its correct version, indicating the corrections needed. It was used as an evaluation metric in PDDL domain model generation. We compare the generated HPDL model with the correct model and count instances of issues. Each identified issue is counted as one.

### 6.1.2 Hierarchy Evaluation

**Count of Hierarchy Issues:** This includes:

1. **Correct task sequences:** Verifying that task sequences in each method are logical and complete.
2. **Removal of irrelevant preconditions in methods:** Identifying and eliminating unnecessary preconditions from methods.
3. **Addition of missing precondition in methods:** Ensuring all necessary preconditions are present in methods.
4. **Inclusion of missing subtasks inside tasks:** Confirms that all necessary subtasks (can be actions or other tasks) are included in the task sequence inside the method to achieve the goal of the parent task.
5. **Removal of irrelevant subtasks inside tasks:** Remove the unnecessary subtasks (can be actions or other tasks) from the methods.

This metric evaluates the hierarchical features of HPDL, ensuring the task network's structure is coherent, efficient, and logical. Each identified issue is counted as one.

### 6.1.3 Basic Syntax Error Evaluation

**Count of Basic Syntax Error:** This includes:

1. **Unique and intuitive names:** Ensuring each predicate has a unique and intuitive name with clearly defined parameters.
2. **Correct typing of parameters and return values:** Verifying that parameters and return values are correctly typed.
3. **Correction of typos or spelling mistakes:** Identifying and correcting errors.
4. **Balanced parentheses:** Ensuring all parentheses are balanced.
5. **Consistent naming conventions:** Maintaining consistent naming conventions across the domain model.

### 6.1.4 Total Error Value Calculation

The total error value calculation formula integrates three distinct evaluation metrics to provide an assessment of the generated HPDL models. A higher total error value indicates a greater distance between the generated HPDL model and its corrected version. Corrected versions are those versions that can generate plans. Each identified issue in any category is counted as one, reflecting its impact on the overall quality of the model.

The weights assigned to each category reflect the relative importance of each evaluation category, with hierarchy errors being the most critical and basic syntax errors the least.

Each type of evaluation contributes differently to the total error value:

1. **General Evaluation:** This metric, weighted at 1, assesses the basic elements of the HPDL model, such as the inclusion and correctness of preconditions, effects, and parameters. It identifies fundamental issues that affect the model’s basic functionality and usability, ensuring that the generated model aligns closely with the expected output.
2. **Hierarchy Evaluation:** This metric, weighted at 1.5, focuses on the hierarchical structure of the task networks. It evaluates the logical sequencing of tasks, method preconditions, and subtask coverage. This metric has a higher weight because correct task decomposition and hierarchical consistency are highly important for AI planning.
3. **Basic Syntax Error Evaluation:** This metric, weighted at 0.5, checks for syntactical correctness, including unique naming, parameter typing, and balanced parentheses. Although syntax errors hinder the process of parsing and executing code, they are easier to fix compared to structural or logical defects. Therefore, we are assigning a lesser weight to syntax errors when calculating the total error value.

By assigning different weights to these metrics, Formula 6.1 prioritizes hierarchical consistency and task logic over basic syntax corrections, thereby aligning the error evaluation with the practical importance of each aspect in ensuring correct and efficient HPDL models.

Below is the formula for the Total Error Value:

$$(6.1) \text{ Total Error Value} = (1 \times GE) + (1.5 \times HE) + (0.5 \times BSE)$$

where:

- *GE* denotes the count of General Evaluation
- *HE* denotes the count of Hierarchy Evaluation
- *BSE* denotes the count of Basic Syntax Error Evaluation

The evaluation metrics detailed above will help systematically assess the quality of HPDL domain models generated by LLMs. By focusing on general evaluation, hierarchy evaluation, and basic syntax error evaluation, we can measure the correctness, consistency, and accuracy of the generated models and identify areas for improvement.

## 6.2 Experimental Setup

The experimental setup for evaluating the generation of HPDL models using LLMs involves two primary approaches. The first approach directly uses prompts to generate complete domain models. The second approach uses a structured, chaining method where actions and tasks are generated incrementally. This approach aims to reduce complexity and enhance the accuracy of the generated models by incorporating generated elements into subsequent prompts. The second approach is also performed manually. This section details the experimental setup, including the types of prompt structures used and examples of prompts for action and task generation. Each prompt is executed three times, and then the average of the evaluation metric values is taken from the three domain model outputs.

### 6.2.1 Approaches to Generating HPDL Models

- **Direct Prompt Generation:**

The direct prompt generation approach attempts to create an entire domain model in one pass. This method leverages the LLM’s ability to generate outputs from detailed inputs. However, the complexity of producing a complete and accurate domain model in a single step can be challenging and lead to errors and inconsistencies.

- **Incremental Chaining Method:**

The incremental chaining method addresses the complexity issue by breaking down the task into smaller, manageable parts. This method involves generating actions and tasks sequentially and incorporating each generated element into subsequent prompts. This approach ensures a structured and coherent domain model by starting with the most basic action and progressively building upon it.

**Workflow for Incremental Chaining Method:**

1. **Generate Actions:**

- Generate the first action using a prompt.
- Include the generated action in the prompt for the next action.
- Repeat until all actions are generated.

2. **Generate Tasks:**

- Generate the first task using all previously generated actions.
- Include the generated task in the prompt for the next task.
- Repeat until all tasks are generated.

3. **Combine Actions and Tasks:**

- Combine all generated actions and tasks to form the complete domain model.

### 6.2.2 Prompt Structure

To evaluate the efficiency of both approaches, we used four types of structured prompts. These prompts were designed to provide varying levels of detail and context to the LLMs, influencing the quality of the generated outputs. The prompt structures are as follows:

1. **S1:**

- Basic information about HPDL.
- Sample HPDL domain model, action, or tasks.
- Detailed information about the domain and the tasks or model to be generated.

2. **S2:**

- Similar to S1, but the main task description is in natural language rather than technical terms. Refer to Appendix A S2 for the prompt and Refer to Appendix A Output S2 for the prompt result.
3. **S3:**
    - Similar to S2, but with the positions of the sample and task descriptions swapped.
  4. **S4:**
    - Similar to S2, but without the detailed task description, relying on zero-shot prompting.

These prompts were tested using GPT-3.5, GPT-4, and GPT-4o to evaluate their performance across different LLMs. This evaluation aims to identify the most effective strategies for leveraging LLMs in AI planning, ensuring the generated models are correct, consistent, complete and accurate for practical for real-world applications.

## 6.3 Model Quality Assessment

In this section, we evaluate the quality of the HPDL models generated using the experimental approaches in detail. The assessment is based on the values of the total error value which is based on different evaluation metrics, such as General Evaluation, Hierarchy Evaluation, and Basic Syntax Error Evaluation. A higher total error value indicates a greater deviation from the ideal model which is capable of generating correct plans, whereas a lower total error value indicates a better and more accurate model.

The results obtained from GPT-4 and GPT-4o have been discussed as they provide the most accurate results. The results from GPT-3.5 were below par and had errors. Additionally, the results of Structure 4 (S4) have not been discussed as they had numerous errors and noise, making it difficult to calculate error values accurately. One major reason for this inaccuracy is that GPT had limited prior knowledge of HPDL and its syntax, resulting in a significant amount of noise and errors. In contrast, the other structures that used n-shot prompting techniques gave better results. These prompts were generated with HPDL examples, helping the LLMs learn the syntax and generate HPDL accordingly.

Contrary to expectations, the sequential generation strategy did not significantly outperform the complete domain model generation approach. Both techniques yielded similar results, indicating that while breaking down tasks into smaller parts might seem beneficial, the overall effectiveness was comparable to generating the entire model in one go.

We discuss each of the four domains one by one, starting with the smallest and least complex domains, which have the fewest actions and tasks followed by the other domains in order of complexity.

### 6.3.1 Basic Domain

The basic domain from the JSHOP2 distribution serves as a fundamental testing ground for evaluating HPDL models. This domain involves simple tasks and actions, making it an ideal starting point for assessing the capabilities of LLMs in generating planning models. The domain includes two primary actions: pick up an object and drop an object, and one task: swap the objects.

#### Generated Model Evaluation

The HPDL models generated for the basic domain were highly accurate, with all models correctly defining the necessary actions and tasks. Each model accurately represented the logical structure and details required for the domain as seen in Table 6.4. All generated models included the required actions and tasks and covered the domain’s needs without any missing elements. The simplicity of the domain contributed to this completeness. The models maintained excellent coherence, with no contradictions or inconsistencies in the naming conventions for actions, predicates, and tasks as there were no syntax errors.

#### Plan Generation

- **Success Rate:** All generated models successfully produced valid plans in the SH planner. The plans correctly executed the pick-up and drop actions to achieve the task of swapping objects.
- **Error Analysis:** The Total Error Value was 0 in all cases, indicating perfect correctness and accuracy. This was consistent across all three prompt structures (P1, P2, P3) and for both GPT-4o and GPT-4.

The basic domain’s simplicity and small scale provided an ideal environment for initial tests. The generated models from all prompt structures and both GPT-4 and GPT-4o demonstrated perfect correctness, completeness, and consistency. The total error score of 0 in all cases indicates that the models were highly accurate and successful in generating valid plans in the SH planner. These positive results highlight the potential for using LLMs in more complex domains, given their success in this foundational test.

Evaluation Metrics	GPT-4o			GPT-4		
	S1	S2	S3	S1	S2	S3
<b>General Evaluation</b>	0	0	0	0	0	0
<b>Hierarchy Evaluation</b>	3.5	5	3.5	3.5	5	3.5
<b>Basic Syntax Errors</b>	0.5	2	1	0	3	0.5
<b>Total Error Value</b>	5.5	8.5	6	5	9	5.5

**Table 6.1:** Complete evaluation metrics values for GPT-4o and GPT-4 in the blocks world domain for prompt structures S1, S2, and S3.

### 6.3.2 Blocks World Domain

The blocks world domain is a classic AI planning problem involving the manipulation of blocks to achieve a specified goal configuration. This domain is moderately complex, requiring the generation of four actions and four tasks. The actions include stacking, unstacking, moving, and placing blocks, while the tasks involve various combinations of these actions to achieve desired configurations. Refer to Table 6.1 for the detailed values of evaluation metrics.

#### Generated Model Evaluation

The HPDL models generated by GPT-4 and GPT-4o for the blocks world domain were mostly complete as they included all required actions, tasks, methods, and predicates, indicating that the LLMs were able to understand and encapsulate the domain's essential components effectively. The models were consistent, with no contradictions and a uniform naming convention for actions, predicates, and tasks, ensuring logical coherence.

However, the models were not fully accurate as seen in Table 6.4, when assessed using the Total Error Value. The values ranged between 5 to 10 depending on the prompt structures and the LLM used. This indicates that while the models captured the overall structure of the domain, they had several inaccuracies and omissions.

One significant error observed was the use of “exists” in the preconditions of methods. Initially, it was perceived that this error was due to the presence of the word “exists” in the prompts. However, even after the removal of the word from the prompts, the models continued to use “exists” as a syntax in the preconditions. This likely occurred because “exists” is a valid syntax in PDDL, which the GPT models are familiar with, but not in HPDL. This highlighted a Domain-Specific Knowledge Gap, as the model knows about PDDL but not HPDL, leading to incorrect assumptions or applications of syntax. Additionally, this issue is an example of where knowledge from one domain (PDDL) interferes with or incorrectly influences the understanding by generating information in another domain (HPDL). “exists” being used from (PDDL) syntax is an assumption, it could have been picked from any other domain language or any other language syntax as well. Despite these issues, the generated actions and task sequences in methods were generally correct and in proper order.

The other errors were predominantly found in the task sequence of methods inside tasks (Table 6.1). Additionally, there were some instances where the LLM confused predicates and tasks due to similar meanings of task and predicates and naming conventions. For example, the predicate “clear” and the task “clear-block” were often confused. This confusion is a result of both the predicate and task having meanings that are quite similar. Predicate “clear” meant that the lock had no block on top of it whereas the task “clear-block” was to clear if there was any block on top. Initially, it was assumed that this could be because of the use of similar names but it was also observed for other predicates and tasks where the names were different but had similar meanings.

#### Plan Generation

- **Success Rate:** The models had a mixed success rate in producing valid plans due to the mentioned errors. The inclusion of incorrect preconditions and the confusion between predicates and tasks affected the plan generation's effectiveness.

- **Error Analysis:** The Total Error Value, ranging from 5 to 10 as mentioned in Table 6.4, reflects these issues. The primary errors involved incorrect preconditions in methods and the misapplication of syntax and task sequences.
- **Correctness and Accuracy:** The Total Error Value indicated that the models were not perfectly accurate. The use of PDDL syntax like “exists” and the confusion between similarly named predicates and tasks were significant sources of error.

The blocks world domain presented a moderate level of complexity, and the generated HPDL models from GPT-4 and GPT-4o showed varying degrees of success. While the models were complete and consistent, they fell short in terms of correctness and accuracy, as indicated by the Total Error Values. The primary issues involved the incorrect use of syntax and confusion between predicates and tasks with similar meanings. Addressing these problems is crucial for improving the LLM’s ability to generate accurate and reliable models for more complex domains. Overall, the blocks world domain provided valuable insights into the strengths and limitations of LLMs in generating HPDL models for moderately complex planning tasks.

Evaluation Metrics	GPT-4o			GPT-4		
	S1	S2	S3	S1	S2	S3
<b>General Evaluation</b>	1	1.5	1	0	1	1
<b>Hierarchy Evaluation</b>	4.5	6	5.5	6.5	6.5	6
<b>Basic Syntax Errors</b>	5.5	5.5	5	5	5.5	5
<b>Total Error Value</b>	10.5	13	12	12	13.5	13

**Table 6.2:** Complete evaluation metrics values for GPT-4o and GPT-4 in the satellite domain for prompt structures S1, S2, and S3.

### 6.3.3 Satellite Domain

The satellite domain involves coordinating multiple satellites to manage tasks such as capturing images, adjusting satellite positions, and calibrating the instruments. This domain includes five actions and six tasks, making it slightly more complex than the blocks world domain due to its hierarchical structure and the interdependence of tasks and actions. Refer to Table 6.2 for the detailed values of evaluation metrics.

#### Generated Model Evaluation

The HPDL models generated by GPT-4 and GPT-4o for the satellite domain were mostly complete as they included all required actions, tasks, methods, and predicates, indicating that the LLMs were able to understand and encapsulate the domain’s essential components effectively. The models were consistent, with no contradictions and a uniform naming convention for actions, predicates, and tasks, ensuring logical coherence.

However, the models were not fully accurate or correct when assessed using the Total Error Value as observed in Table 6.4, which ranged between 10 to 15 depending on the prompt structures and the LLM used. This indicates that while the models captured the overall structure of the domain, they had several inaccuracies and omissions.

One significant error observed was the use of “exists” in the preconditions of methods. This issue, similar to the blocks world domain, likely arises from the model’s familiarity with PDDL, where “exists” is a valid syntax, leading to a Syntax error. This demonstrates a Domain-Specific Knowledge Gap where the model applies PDDL syntax incorrectly to HPDL. “exists” is being used from PDDL syntax is an assumption, it could have been picked from any other domain language or any other language syntax.

Most errors were found in the preconditions of methods. The LLMs struggled to accurately interpret or generate the necessary preconditions, likely due to the complexity and hierarchical nature of the tasks. This increased complexity compared to the blocks world domain and made it more challenging for the models to handle the complicated dependencies and preconditions required for correct task specification.

Despite these issues, there were no significant errors in the task sequences of the methods inside the tasks. This indicates that while the preconditions were problematic, the overall sequence and logical flow of tasks were correctly generated. The hierarchical structure of the satellite domain contributed to these issues, as the more complex dependencies required a more refined understanding and accurate representation.

### Plan Generation

- **Success Rate:** The models had a mixed success rate in producing valid plans, largely due to the errors in preconditions. However, the correct task sequences helped mitigate some of these issues, allowing for more successful plan generation than might be expected from the Total Error Value.
- **Error Analysis:** The Total Error Value, ranging from 10 to 15, reflects the higher complexity and the number of inaccuracies in the generated models. The primary errors were encountered in the preconditions of methods, where the LLMs failed to accurately capture the necessary conditions for task execution.
- **Correctness and Accuracy:** While the Total Error Value was higher than in the blocks world domain, indicating more inaccuracies, the models’ performance in maintaining correct task sequences suggests that they were on par with or slightly better than the blocks world models in handling the overall task structure as there were no errors in the task sequence.

The larger size of the satellite domain, with more actions and tasks, led to a higher error value. This does not indicate poorer performance but rather reflects the increased complexity and the challenges associated with accurately modelling such a domain.

The satellite domain, with its increased complexity and hierarchical nature, provided a challenging test for the LLMs. The generated HPDL models from GPT-4 and GPT-4o were mostly complete and consistent but had significant inaccuracies in the preconditions of methods, leading to higher Total Error Values. Despite these issues, the correct task sequences and logical coherence of the

models indicate a reasonable understanding and capability in handling the domain. The results suggest that while the models for the satellite domain had a higher error rate, they were on par with or better than those for the blocks world domain in terms of overall structure and sequence accuracy. This highlights the potential of LLMs to generate complex planning models with further refinement and better handling of domain-specific syntax and dependencies.

Evaluation Metrics	GPT-4o			GPT-4		
	S1	S2	S3	S1	S2	S3
<b>General Evaluation</b>	0	1	0	0	0	0
<b>Hierarchy Evaluation</b>	45	50	47	43	52	46
<b>Basic Syntax Errors</b>	25	34	28	25	36	26
<b>Total Error Value</b>	80	93	84.5	77	96	82

**Table 6.3:** Complete evaluation metrics values for GPT-4o and GPT-4 in the cafeteria domain for prompt structures S1, S2, and S3.

### 6.3.4 Cafeteria Domain

The cafeteria domain is the most complex and hierarchical of all the domains tested. It involves numerous actions, tasks, methods, and predicates to manage various scenarios within a cafeteria setting. Tasks include adjusting lighting, handling multiple environmental conditions, and coordinating various automated systems. The complexity and deep hierarchical nature of this domain makes it a challenging test for the capabilities of LLMs. Refer to table 6.3 for the detailed values of evaluation metrics.

#### Generated Model Evaluation

The HPDL models generated by GPT-4 and GPT-4o for the cafeteria domain were mostly complete, including nearly all required actions, tasks, methods, and predicates. However, a few extra tasks were introduced that were not part of the original specification. This indicates a tendency of the LLMs to overgenerate components, possibly due to the complex nature of the domain.

One significant issue observed was the absence of preconditions for some actions, where the precondition tag was missing. This was a syntax error that resulted in incomplete action definitions. This lack of precondition tags resulted in a deviation from the previously observed behavior in simpler domains where all actions had preconditions.

Despite these issues, the models were consistent, with no contradictions and a uniform naming convention for actions, predicates, and tasks. This consistency ensured logical coherence within the models, although it did not fully compensate for the other inaccuracies.

Similar to the blocks world and satellite domains, the issue of “exists” keyword in preconditions of methods was observed in this domain as well. Additionally, the deep hierarchical structure of the tasks and actions, combined with almost similar names and meanings, led to numerous errors in task sequences within methods.

Furthermore, while GPT-4 and GPT-4o lack specific knowledge about HPDL and derive all their HPDL-related knowledge from examples or information given in the prompt, they managed to use syntaxes like “Increase” “Decrease” and “Forall” correctly in HPDL. This is likely because these syntaxes are part of PDDL, and HPDL builds on PDDL. Therefore, the use of these syntaxes did not cause errors, demonstrating some overlap in the syntax between PDDL and HPDL.

### Plan Generation

- **Success Rate:** The generated models had a lower success rate in producing valid plans due to the high number of errors in preconditions and task sequences. The deep hierarchical nature of the tasks and actions posed a significant challenge.
- **Error Analysis:** The Total Error Value for this domain ranged between 75 to 95, indicating high complexity and inaccuracies in the generated models. The primary errors included missing preconditions for actions, incorrect preconditions in methods, and numerous errors in task sequences due to similar names and meanings of tasks and actions.
- **Correctness and Accuracy:** The high Total Error Value indicates that the models were not very accurate or correct. The LLMs struggled with the deep hierarchical nature and the complexity of the domain, leading to significant errors.

Interestingly, while the LLMs lacked specific knowledge of HPDL, they correctly used certain syntaxes like “Increase”, “Decrease”, and “Forall” which are shared between PDDL and HPDL. For instance, “Increase” and “Decrease” were used to adjust the light intensity values in actions, and “Forall” was used in preconditions of actions. These correct usages indicate that the LLMs can draw on their knowledge of PDDL to some extent, even though they were not explicitly trained on HPDL.

The cafeteria domain, being the most complex and hierarchical, imposed significant challenges for the LLMs. While the generated HPDL models were mostly complete and consistent, they were far from accurate or correct. The high Total Error Value, ranging from 75 to 95, highlights the difficulty the models had in handling the complexity and deep hierarchical structure of the domain. Issues such as missing preconditions, syntax errors, and numerous errors in task sequences reflected the LLMs’ struggle with this complex domain.

Domain Models	GPT-4o			GPT-4		
	S1	S2	S3	S1	S2	S3
Basic Domain	0	0	0	0	0	0
Blocks world	5.5	8.5	6	5	9	5.5
Satellite	10.5	13	12	12	13.5	13
Cafeteria	80	93	84.5	77	96	82

**Table 6.4:** Comparison of total error value on GPT-4 and GPT-4o across different domain models for various prompt structures used to generate complete domain models.

The successful use of some PDDL syntaxes like “Increase”, “Decrease”, and “Forall” shows the potential for leveraging existing knowledge to improve HPDL model generation. However, the overall performance indicates that significant improvements are needed to handle such complex domains

effectively. The results from the cafeteria domain emphasise the importance of refining prompt structures and enhancing the LLMs' understanding of domain-specific syntax and hierarchical dependencies for better accuracy and reliability in generating complex planning models.

# 7 Conclusion and Outlook

## 7.1 Summary of Findings

This thesis explored the use of LLMs for the automated generation of HPDL models. The thesis showed that LLMs have the ability to create HPDL models and evaluate them based on several metrics, including total error value. These results are then used to establish the accuracy, correctness, completeness, and consistency of these domains. Key findings include:

1. **Effectiveness of LLM-Based Approach:** LLMs, particularly GPT-4 and GPT-4o, showed potential in generating structured HPDL models from natural language prompts. The generated models were coherent with some issues in syntax and precondition accuracy. Very deep hierarchical and complex models faced significant issues, however, there is scope to enhance the prompts more efficiently.
2. **Evaluation Metrics:** The introduced evaluation metrics, including General Evaluation, Hierarchy Evaluation, and Basic Syntax Error Evaluation, provided a complete assessment framework. These metrics highlighted the strengths and weaknesses of the generated models, with hierarchical consistency being the most challenging aspect for the LLMs.
3. **Experimental Results:** The experiments across domains like basic domain, blocks world, and satellite showed that LLMs can handle simple to moderately complex domains effectively. However, the cafeteria domain showed that deep hierarchical structures and domain-specific knowledge gaps led to higher error values in more complex domains. Prompt engineering techniques, such as prompt chaining and n-shot prompting, were beneficial. Detailed examples in the prompts significantly improved the results, while zero-shot models (without examples) were unable to generate results and had lots of errors and noises.
4. **Prompt Engineering:** The prompt engineering techniques used in this research, including prompt chaining and n-shot prompting, played a crucial role in improving the quality of generated models. These techniques helped in breaking down complex tasks into manageable parts, thereby enhancing the quality of generated models. The use of detailed examples in the prompts led to better results. Zero-shot models consistently performed poorly due to lack of examples, emphasizing the importance of example-driven prompts.

## 7.2 Implications for Industry and Research

The findings from this research have significant implications for both industry practitioners and researchers:

1. **Efficiency and Scalability:** The ability of LLMs to generate HPDL models can significantly reduce the time and effort required for manual domain model creation. This can lead to improved efficiency in developing AI planning systems, particularly in domains like robotics, smart buildings, and autonomous vehicles.
2. **Adaptability:** The approach shows promise for adaptability across various domains, suggesting that with further refinement, LLMs could become a standard tool for domain model generation. This adaptability can facilitate the deployment of AI planning systems in diverse industrial applications.
3. **Research Advancements:** For researchers, the evaluation framework and experimental insights provide a foundation for further exploration into improving LLM capabilities in hierarchical planning. The integration of LLMs with classical planning algorithms, as suggested in the literature, could enhance their practical applicability and accuracy.

### 7.3 Limitations

This study acknowledges several limitations:

1. **Availability and Fine-Tuning of GPT APIs:** Access to GPT APIs for fine-tuning is often limited due to high costs and restrictive policies. This restriction reduces the effectiveness of fine-tuning the LLMs based on HPDL syntax as well as domain knowledge. The easy availability of fine-tuning facilities can help generate better domain models.
2. **Syntax Checking for HPDL:** The absence of automated tools or plugins for HPDL syntax checking requires time-consuming and error-prone manual verification. This reduces productivity and increases the potential for overlooked errors, hindering the overall efficiency of the research process.

### 7.4 Recommendations

Based on the research findings, several recommendations can be made to improve the practice of domain model generation in AI planning:

1. **Enhanced Examples in Prompts:** Developing more comprehensive examples in prompts that include examples of HPDL syntax and hierarchical structures can help LLMs better understand and generate accurate models.
2. **Refinement of Prompt Structures:** Refining the prompt structures used for LLMs, as outlined in the experiments, can improve the quality of generated models. Including more context and examples in the prompts can lead to better results.
3. **Tool Development:** Developing user-friendly tools that incorporate LLM-based HPDL generation can make this technology more accessible to practitioners. Such tools could include features for error checking and correction to enhance the usability of generated models.

## 7.5 Future Work

The research conducted in this thesis opens several avenues for future exploration:

1. **HPDL-Specific Fine-Tuning:** Fine-tuning LLMs on HPDL syntax can lead to a better understanding of results and can reduce the syntax-based errors.
2. **Domain-Specific Fine-Tuning:** Further fine-tuning of LLMs on domain-specific corpora can improve their understanding and generation of hierarchical models. This can help address the domain-specific knowledge gaps observed in the experiments. Fine-tuning can also resolve issues related to syntax and hierarchical understanding problems faced by the models.
3. **Error Reduction Strategies:** Investigating strategies to reduce syntax and logical errors in generated models is crucial. Techniques such as in-context learning, iterative prompting, and incorporating feedback loops can be explored.
4. **Empirical Validation:** Conducting empirical validation of LLM-generated models in real-world applications can provide insights into their practical effectiveness and limitations. This can guide further improvements and refinements in the approach.

In conclusion, the research demonstrated the potential of LLMs in generating hierarchical domain models, paving the way for more efficient and scalable AI planning systems. With continued advancements and refinements, LLMs could become integral tools in the development of complex AI planning applications.



## Bibliography

- [AFP+18] A. Arora, H. Fiorino, D. Pellier, M. Métivier, S. Pesty. “A review of learning planning action models”. In: *The Knowledge Engineering Review* 33 (2018). URL: <https://api.semanticscholar.org/CorpusID:56483203> (cit. on p. 32).
- [AGA22] E. Alnazer, I. Georgievski, M. Aiello. “On Bringing HTN Domains Closer to Reality - The Case of Satellite and Rover Domains”. In: *International Conference on Automated Planning Systems (ICAPS) Workshop on Scheduling and Planning Applications (SPARK)*. 2022. URL: [https://icaps22.icaps-conference.org/workshops/SPARK/papers/spark2022\\_paper\\_9.pdf](https://icaps22.icaps-conference.org/workshops/SPARK/papers/spark2022_paper_9.pdf) (cit. on p. 41).
- [AGPA22] E. Alnazer, I. Georgievski, N. Prakash, M. Aiello. “A Role for HTN Planning in Increasing Trust in Autonomous Driving”. In: *IEEE International Smart Cities Conference*. 2022, pp. 1–7. URL: <https://doi.org/10.1109/ISC255366.2022.9922427> (cit. on p. 21).
- [AJO22] D. Aineto, S. Jiménez, E. Onaindía. “A Comprehensive Framework for Learning Declarative Action Models”. In: *J. Artif. Intell. Res.* 74 (2022), pp. 1091–1123. URL: <https://api.semanticscholar.org/CorpusID:250417123> (cit. on p. 32).
- [BMR+20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: <https://arxiv.org/abs/2005.14165> (cit. on p. 25).
- [BSA23] A. Berti, D. Schuster, W. M. P. van der Aalst. *Abstractions, Scenarios, and Prompt Definitions for Process Mining with LLMs: A Case Study*. 2023. arXiv: 2307.02194 [cs.DB] (cit. on p. 16).
- [CGP+23] S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu, S. G. Tiddia. *Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction*. 2023. arXiv: 2307.01128 [cs.CL] (cit. on p. 16).
- [CMW13] S. N. Cresswell, T. L. McCluskey, M. M. West. “Acquiring planning domain models using LOCM”. In: *The Knowledge Engineering Review* 28.2 (2013), pp. 195–213. doi: 10.1017/S0269888912000422 (cit. on p. 16).
- [GA15] I. Georgievski, M. Aiello. “HTN planning: Overview, comparison, and beyond”. In: *Artif. Intell.* 222 (2015), pp. 124–156 (cit. on pp. 15, 21, 22).
- [GB21] I. Georgievski, U. Breitenbücher. “A Vision for Composing, Integrating, and Deploying AI Planning Functionalities”. In: *IEEE International Conference on Service-Oriented System Engineering*. 2021, pp. 166–171 (cit. on p. 15).

- [Gef02] H. Geffner. “Perspectives on artificial intelligence planning”. In: *AAAI/IAAI*. 2002. URL: <https://api.semanticscholar.org/CorpusID:2169674> (cit. on p. 19).
- [Geo] I. Georgievski. *GitHub - PlanX-Universe/sh: The SH Planning System designed to tackle complex planning problems utilizing the principles of HTN planning*. — *github.com*. <https://github.com/PlanX-Universe/sh>. [Accessed 07-07-2024] (cit. on p. 43).
- [Geo13] I. Georgievski. *HPDL: Hierarchical Planning Definition Language*. JBI Preprint 2013-12-3. Uni. of Groningen, 2013. URL: <https://ilche.mk/wp-content/uploads/papers/georgievski2013-hpdl.pdf> (cit. on pp. 15, 23, 42).
- [Geo23] I. Georgievski. “Software Development Lifecycle for Engineering AI Planning Systems”. In: *International Conference on Software Technologies*. 2023, pp. 751–760. DOI: [10.5220/0012149100003538](https://doi.org/10.5220/0012149100003538) (cit. on p. 15).
- [Geo24] I. Georgievski. *sh/src/main/resources/repository at main · PlanX-Universe/sh — github.com*. <https://github.com/PlanX-Universe/sh/tree/main/src/main/resources/repository>. [Accessed 07-07-2024]. 2024 (cit. on pp. 41, 42).
- [GFC13] A. González-Ferrer, J. Fernández-Olivares, L. Castillo. “From business process models to hierarchical task network planning domains”. In: *The Knowledge Engineering Review* 28.2 (2013), pp. 175–193. DOI: [10.1017/S0269888912000410](https://doi.org/10.1017/S0269888912000410) (cit. on p. 22).
- [GK24] V. Gadesha, E. Kavlakoglu. *What is prompt chaining? | IBM — ibm.com*. <https://www.ibm.com/topics/prompt-chaining>. [Accessed 07-07-2024]. 2024 (cit. on p. 26).
- [GNLA17] I. Georgievski, F. Nizamic, A. Lazovik, M. Aiello. “Cloud Ready Applications Composed via HTN Planning”. In: *IEEE International Conference on Service Oriented Computing and Applications*. SOCA’17. 2017, pp. 23–33 (cit. on p. 21).
- [GNN+17] I. Georgievski, T. A. Nguyen, F. Nizamic, B. Setz, A. Lazovik, M. Aiello. “Planning meets activity recognition: Service coordination for intelligent buildings”. In: *Pervasive and Mobile Computing* 38.1 (2017), pp. 110–139 (cit. on pp. 21, 42).
- [GVSK23] L. Guan, K. Valmееkam, S. Sreedharan, S. Kambhampati. *Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning*. 2023. arXiv: [2305.14909](https://arxiv.org/abs/2305.14909) [cs.AI] (cit. on pp. 16, 29, 36, 51).
- [HBB+19] D. Höller, G. Behnke, P. Bercher, S. Biundo, H. Fiorino, D. Pellier, R. Alford. *HDDL – A Language to Describe Hierarchical Planning Problems*. 2019. arXiv: [1911.05499](https://arxiv.org/abs/1911.05499) [cs.AI]. URL: <https://arxiv.org/abs/1911.05499> (cit. on p. 43).
- [HTP+20] Y. Huo, J. Tang, Y. Pan, Y.-f. Zeng, L. Cao. “Learning a Planning Domain Model From Natural Language Process Manuals”. In: *IEEE Access* 8 (2020), pp. 143219–143232. URL: <https://api.semanticscholar.org/CorpusID:221159550> (cit. on p. 32).
- [Ign22] IgnacioVellido. *GitHub - UGR-IntelligentSystemsGroup/HPDL-Planner: SIADEX - An HTN planner with temporal, partial order planning — github.com*. <https://github.com/UGR-IntelligentSystemsGroup/HPDL-Planner>. [Accessed 07-07-2024]. 2022 (cit. on p. 43).

- [Jil20] R. Jilani. “Automated Domain Model Learning Tools for Planning”. In: 2020. URL: <https://api.semanticscholar.org/CorpusID:216191576> (cit. on p. 16).
- [KBK+23] N. Klietsova, J.-V. Benzin, T. Kampik, J. Mangler, S. Rinderle-Ma. *Conversational Process Modelling: State of the Art, Applications, and Implications in Practice*. 2023. arXiv: 2304.11065 [cs.CL] (cit. on p. 16).
- [LJZ+23] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, P. Stone. *LLM+P: Empowering Large Language Models with Optimal Planning Proficiency*. 2023. arXiv: 2304.11477 (cit. on p. 29).
- [LLL+22] J. Liu, A. Liu, X. Lu, S. Welleck, P. West, R. L. Bras, Y. Choi, H. Hajishirzi. *Generated Knowledge Prompting for Commonsense Reasoning*. 2022. arXiv: 2110.08387 [cs.CL]. URL: <https://arxiv.org/abs/2110.08387> (cit. on pp. 26, 27).
- [LPK+24a] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, M. Aiello. *DELTA: Decomposed Efficient Long-Term Robot Task Planning using Large Language Models*. 2024. arXiv: 2404.03275 [cs.R0]. URL: <https://arxiv.org/abs/2404.03275> (cit. on p. 16).
- [LPK+24b] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, M. Aiello. *Towards Human Awareness in Robot Task Planning with Large Language Models*. 2024. arXiv: 2404.11267 [cs.R0]. URL: <https://arxiv.org/abs/2404.11267> (cit. on p. 16).
- [MSF+23] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, M. Martin. *LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT*. 2023. arXiv: 2307.06917 [cs.AI] (cit. on p. 16).
- [MVV16] T. L. McCluskey, T. Vaquero, M. Vallati. “Issues in Planning Domain Model Engineering”. In: *PlanSIG*. 2016 (cit. on pp. 16, 22).
- [RWC+19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. “Language Models are Unsupervised Multitask Learners”. In: 2019. URL: <https://api.semanticscholar.org/CorpusID:160025533> (cit. on p. 24).
- [SDS+23] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, M. Katz. *Generalized Planning in PDDL Domains with Pretrained Large Language Models*. 2023. arXiv: 2305.11014 [cs.AI]. URL: <https://arxiv.org/abs/2305.11014> (cit. on p. 31).
- [VSM+23] K. Valmeekam, S. Sreedharan, M. Marquez, A. Olmo, S. Kambhampati. *On the Planning Abilities of Large Language Models (A Critical Investigation with a Proposed Benchmark)*. 2023. arXiv: 2302.06706 [cs.AI]. URL: <https://arxiv.org/abs/2302.06706> (cit. on p. 30).
- [WWS+23a] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023 (cit. on p. 26).
- [WWS+23b] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: <https://arxiv.org/abs/2201.11903> (cit. on p. 27).
- [XJL+17] X. Xu, R. Ju, X. Liu, G. Li, Y. J. Son. “Extending HTN to planning and execution control for small combat unit simulation”. In: *Int. J. Model. Simul. Sci. Comput.* 8 (2017), 1750032:1–1750032:18. URL: <https://api.semanticscholar.org/CorpusID:53744540> (cit. on p. 22).

## Bibliography

---

- [XYJ+23] D. Xie, B. Yoo, N. Jiang, M. Kim, L. Tan, X. Zhang, J. S. Lee. *Impact of Large Language Models on Generating Software Specifications*. 2023. arXiv: [2306.03324](https://arxiv.org/abs/2306.03324) [cs.SE] (cit. on pp. 16, 30).
- [XYZ+23] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, H. Soh. *Translating Natural Language to Planning Goals with Large-Language Models*. 2023. arXiv: [2302.05128](https://arxiv.org/abs/2302.05128) [cs.CL] (cit. on pp. 16, 36).
- [ZHY+09] H. H. Zhuo, D. H. Hu, Q. Yang, H. Muñoz-Avila, C. Hogg. “Learning applicability conditions in AI planning from partial observations”. In: *International Joint Conference on Artificial Intelligence*. 2009. URL: <https://api.semanticscholar.org/CorpusID:9924248> (cit. on p. 16).
- [ZZL+10] J. Zhang, L. Zhang, X. Li, Y. Hu, J. Zhang. “Integrating HTN planner in cleaning-security robot: Handling planning with memory and problem template”. In: *2010 IEEE International Conference on Automation and Logistics* (2010), pp. 55–60. URL: <https://api.semanticscholar.org/CorpusID:13539260> (cit. on p. 22).

# A Appendix

## P1

### P1: Basic Information about HPDL

Generate HPDL domain model for the given information. These are the basic information and structure about HPDL and its syntax. Information about the domain, types, predicates, functions, actions, and tasks will be provided in the domain description.

#### General Guidelines:

- **Preconditions and Effects:** Individual conditions in preconditions and effects should be listed separately. For example, “lamp is checked - and turned on” should be written as two separate conditions: “lamp is checked” and “lamp is turned on”.
- **Unique Predicate Names:** Each predicate in HPDL must have a unique name. Ensure that predicate names are intuitive and readable, and their parameters are explicitly defined.
- **Action Definitions:** For each action, define the parameters, preconditions, and effects. Ensure that preconditions and effects are logically consistent and relevant to the action being described.
- **Task Definitions:** For each task, define the parameters and methods. Methods should provide alternative ways to achieve the task, with each method having preconditions and tasks. These tasks inside the methods are subtasks, which are logically ordered (e.g., using sequence). Tasks inside methods can consist of actions as well as tasks.

#### Components of HPDL:

- **Domain Definition:** Begin with the domain name and required features.
- **Types:** Define the types of objects used in the domain.
- **Predicates:** List predicates that describe the properties of objects or relationships between them.
- **Functions:** Define any numeric functions used to represent quantitative aspects of the domain.
- **Actions:** Define actions with parameters, preconditions, and effects.
- **Tasks:** Define high-level tasks with parameters and methods.

- **Methods:** Methods provide alternative ways to achieve the task. Each method has preconditions and tasks. These tasks inside the methods are subtasks, which are logically ordered (e.g., using sequence). Tasks inside methods can consist of actions as well as other tasks.

## P2

### P2: Sample HPDL Domain Model

Here are examples from the classical blocks world domain to demonstrate the output format for the generation of action:

**Domain Information:** Blocks world is a planning domain in AI. The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved. There is only one type of object in this domain, and that is the block.

This 'blocks world' domain is a representation of the classical planning domain for the blocks world.

- The objective is to construct one or more stacks of blocks.
- During this process, only one block may be moved at any given time,
- with options to place it either on the table or on top of another block.
- Blocks that are under another block cannot be moved.
- A robotic arm is used for manipulating a block.

**Example 1: Action:** This action enables the robot to put a block onto the table. For example, the robot puts block\_1 onto the table.

You can create and define new predicates, but you may also reuse the following predicates:

- (on ?x - block ?y - block)
- (ontable ?x - block)
- (clear ?x - block)
- (handempty)
- (holding ?x - block)

For this objective, the desired HPDL model for the action "Put Down the Block" is something like this:

```
(:action put-down
  :parameters (?x - block)
  :precondition (holding ?x)
  :effect (and (not (holding ?x)) (clear ?x) (handempty) (ontable ?x))
)
```

---

**Example 2: Action:** This action enables the robot to put a block on top of another block. For example, the robotic arm places block x on top of block y if the arm is holding block x and block y is clear.

You can create and define new predicates, but you may also reuse the following predicates:

- (on ?x - block ?y - block)
- (ontable ?x - block)
- (clear ?x - block)
- (handempty)
- (holding ?x - block)

For this objective, the desired HPDL model for the action SStack the Block is something like this:

```
(:action stack
  :parameters (?x - block ?y - block)
  :precondition (and (holding ?x) (clear ?y))
  :effect (and (not (holding ?x)) (not (clear ?y)) (clear ?x) (handempty) (on ?x ?y))
)
```

## P3

### P3: Detailed Domain and Task Information

Here are the Details of the domain for which you need to generate the action 'Take an image' for the HPDL Domain model:

The domain represents a scenario where satellites equipped with various instruments are used to capture images of specific directions. It includes actions to turn the instruments on and off, calibrate them, and point the satellite in the correct direction. The domain supports hierarchical planning with compound tasks and methods to achieve these tasks.

**Overview:**

- The domain aims to ensure successful image capture by managing the power and calibration status of instruments according to the satellite's direction.
- There are four types of objects in this domain: satellite, direction, instrument, and mode.
- The object type instrument includes various instruments onboard the satellites that need to be powered on and calibrated before use.
- The object type direction represents the different directions in which a satellite can be pointed.

- The object type mode represents the different modes in which an instrument can operate.
- The primary tasks the domain handles include turning instruments on or off, calibrating instruments, pointing satellites, and capturing images.
- These tasks ensure that the satellite operations are performed efficiently and accurately to achieve the mission objectives.

**Required features:**

- strips
- typing
- negative-preconditions
- equality

**Types:**

- satellite: Represents the satellite in the domain.
- direction: Represents the direction in which the satellite can point.
- instrument: Represents the instruments onboard the satellite.
- mode: Represents the different modes in which an instrument can operate.

**Predicates:**

- An instrument is onboard a satellite: (on\_board ?i - instrument ?s - satellite)
- An instrument supports a mode: (supports ?i - instrument ?m - mode)
- A satellite is pointing in a direction: (pointing ?s - satellite ?d - direction)
- A satellite has available power: (power\_avail ?s - satellite)
- An instrument is powered on: (power\_on ?i - instrument)
- An instrument is calibrated: (calibrated ?i - instrument)
- An image has been taken of a direction in a mode: (have\_image ?d - direction ?m - mode)
- A direction is a calibration target for an instrument: (calibration\_target ?i - instrument ?d - direction)

**Description of Action to generate:**

**Take an image:**

- This action enables the satellite to take an image in a specific mode if the instrument is calibrated and the satellite is pointing in the correct direction.

- Parameters: satellite (type: satellite), direction (type: direction), instrument (type: instrument), mode (type: mode).
- Precondition: The instrument is calibrated, onboard the satellite, supports the mode, is powered on, and the satellite is pointing in the direction.
- Effect: An image is taken of the direction in the specified mode.

## P4

### P4: Natural Language Variant of P3

Here are the Details of the domain for which you need to generate the action 'Take an image' for the HPDL Domain model:

The domain represents a scenario where satellites equipped with various instruments are used to capture images of specific directions. It includes actions to turn the instruments on and off, calibrate them, and point the satellite in the correct direction. The domain supports hierarchical planning with compound tasks and methods to achieve these tasks.

#### Overview:

- The domain aims to ensure successful image capture by managing the power and calibration status of instruments according to the satellite's direction.
- There are four types of objects in this domain: satellite, direction, instrument, and mode.
- The object type instrument includes various instruments onboard the satellites that need to be powered on and calibrated before use.
- The object type direction represents the different directions in which a satellite can be pointed.
- The object type mode represents the different modes in which an instrument can operate.
- The primary tasks the domain handles include turning instruments on or off, calibrating instruments, pointing satellites, and capturing images.
- These tasks ensure that the satellite operations are performed efficiently and accurately to achieve the mission objectives.

#### Required features:

- strips
- typing
- negative-preconditions

- equality

**Types:**

- satellite: Represents the satellite in the domain.
- direction: Represents the direction in which the satellite can point.
- instrument: Represents the instruments onboard the satellite.
- mode: Represents the different modes in which an instrument can operate.

**Predicates:**

- An instrument is onboard a satellite: (on\_board ?i - instrument ?s - satellite)
- An instrument supports a mode: (supports ?i - instrument ?m - mode)
- A satellite is pointing in a direction: (pointing ?s - satellite ?d - direction)
- A satellite has available power: (power\_avail ?s - satellite)
- An instrument is powered on: (power\_on ?i - instrument)
- An instrument is calibrated: (calibrated ?i - instrument)
- An image has been taken of a direction in a mode: (have\_image ?d - direction ?m - mode)
- A direction is a calibration target for an instrument: (calibration\_target ?i - instrument ?d - direction)

**Description of Action to generate:**

**Take an image:** This action allows the satellite to take an image in a specific mode if the instrument is calibrated and the satellite is pointing in the correct direction. The parameters involved are the satellite, the direction, the instrument, and the mode. The precondition for this action is that the instrument must be calibrated, onboard the satellite, support the mode, be powered on, and the satellite must be pointing in the direction. The effect of the action is that an image will be taken in the direction of the specified mode.

## S2

Instructions for the HPDL Domain Generation Task for basic domain using prompt structure S2

Generate HPDL domain model for the given information. These are the basic information and structure about HPDL and its syntax. Information about the domain, types, predicates, functions, actions, and tasks will be provided in the domain description.

**General Guidelines:**

- **Preconditions and Effects:** Individual conditions in preconditions and effects should be listed separately. For example, “lamp is checked - and turned on” should be written as two separate conditions: “lamp is checked” and “lamp is turned on”.
- **Unique Predicate Names:** Each predicate in HPDL must have a unique name. Ensure that predicate names are intuitive and readable, and their parameters are explicitly defined.
- **Action Definitions:** For each action, define the parameters, preconditions, and effects. Ensure that preconditions and effects are logically consistent and relevant to the action being described.
- **Task Definitions:** For each task, define the parameters and methods. Methods should provide alternative ways to achieve the task, with each method having preconditions and tasks. These tasks inside the methods are subtasks, which are logically ordered (e.g., using sequence). Tasks inside methods can consist of actions as well as tasks.

#### **Components of HPDL:**

- **Domain Definition:** Begin with the domain name and required features.
- **Types:** Define the types of objects used in the domain.
- **Predicates:** List predicates that describe the properties of objects or relationships between them.
- **Functions:** Define any numeric functions used to represent quantitative aspects of the domain.
- **Actions:** Define actions with parameters, preconditions, and effects.
- **Tasks:** Define high-level tasks with parameters and methods.
- **Methods:** Methods provide alternative ways to achieve the task. Each method has preconditions and tasks. These tasks inside the methods are subtasks, which are logically ordered (e.g., using sequence). Tasks inside methods can consist of actions as well as other tasks.

#### **Example from the Classical Blocks World Domain:**

**Domain information:** Blocks world is a planning domain in AI. The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved. There is only one type of object in this domain, and that is the block.

This 'blocks world' domain is a representation of the classical planning domain for the blocks world.

- The objective is to construct one or more stacks of blocks.
- During this process, only one block may be moved at any given time,

- with options to place it either on the table or on top of another block.
- Blocks that are under another block cannot be moved.
- A robotic arm is used for manipulating a block.

For this objective, the desired HPDL Domain Model is something like this:

```
(define (domain blocksworld)
  (:requirements :negative-preconditions :typing)
  (:types block)
  (:predicates
    (on ?x - block ?y - block)
    (ontable ?x - block)
    (clear ?x - block)
    (handempty)
    (holding ?x - block)
  )

  (:action pick-up
    :parameters (?x - block)
    :precondition (and (clear ?x) (ontable ?x) (handempty))
    :effect (and (not (ontable ?x)) (not (clear ?x)) (not (handempty)) (holding ?x))
  )

  (:action put-down
    :parameters (?x - block)
    :precondition (holding ?x)
    :effect (and (not (holding ?x)) (clear ?x) (handempty) (ontable ?x))
  )

  (:action stack
    :parameters (?x - block ?y - block)
    :precondition (and (holding ?x) (clear ?y))
    :effect (and (not (holding ?x)) (not (clear ?y)) (clear ?x) (handempty) (on ?x ?y))
  )

  (:action unstack
    :parameters (?x - block ?y - block)
    :precondition (and (on ?x ?y) (clear ?x) (handempty))
    :effect (and (holding ?x) (clear ?y) (not (clear ?x)) (not (handempty)) (not (on ?x ?y)))
  )

  (:action bk-nop
    :parameters ()
    :precondition ()
    :effect ()
  )
)
```

```

)

(:task do_put_on
  :parameters (?x - block ?y - block)
  (:method m0_do_put_on
    :precondition (and (on ?x ?y))
    :tasks (sequence (bk-nop))
  )
  (:method m1_do_put_on
    :precondition (and (handempty))
    :tasks (sequence (do_clear ?x) (do_clear ?y) (do_on_table ?y) (do_move ?x ?y))
  )
)

(:task do_on_table
  :parameters (?x - block)
  (:method m3_do_on_table
    :precondition (and (clear ?x))
    :tasks (sequence (bk-nop))
  )
  (:method m2_do_on_table
    :precondition (and (clear ?x) (handempty) (not (ontable ?x)) (block ?y) (on ?x ?y))
    :tasks (sequence (unstack ?x ?y) (put-down ?x))
  )
)

(:task do_move
  :parameters (?x - block ?y - block)
  (:method m4_do_move
    :precondition (and (clear ?x) (clear ?y) (handempty) (ontable ?x))
    :tasks (sequence (pick-up ?x) (stack ?x ?y))
  )
  (:method m5_do_move
    :precondition (and (clear ?x) (clear ?y) (handempty) (not (ontable ?x)) (block ?z))
    :tasks (sequence (unstack ?x ?z) (stack ?x ?y))
  )
)

(:task do_clear
  :parameters (?x - block)
  (:method m6_do_clear
    :precondition (and (clear ?x))
    :tasks (sequence (bk-nop))
  )
  (:method m7_do_clear

```

```

:precondition (and (not (clear ?x)) (block ?y) (on ?y ?x) (handempty))
:tasks (sequence (do_clear ?y) (unstack ?y ?x) (put-down ?y))
)
)
)

```

### Details of the Domain for which you need to generate the AI Domain Model:

#### Overview:

- The domain represents a basic scenario where objects (things) are managed by picking them up and dropping them. It includes actions to pick up and drop objects and a compound task to swap two objects. The domain supports hierarchical planning with compound tasks and methods to achieve these tasks.
- The domain aims to demonstrate fundamental features of hierarchical task planning, including simple pick-up and drop actions.
- There is one type of object in this domain: thing.

#### Required features:

- strips
- typing

#### Types:

- thing: A generic type representing any object that can be picked up or dropped.

#### Predicates:

- (have ?t - thing): Indicates that an agent has the specified thing.

#### Actions:

**Pickup a thing:** This action enables the AI to pick up an object. The AI will need to specify the object to be picked up, referred to as the pickup thing. There are no preconditions required for this action. The effect of this action is that the agent will possess the specified thing.

**Drop a thing:** This action enables the AI to drop an object. The AI will need to specify the object to be dropped, referred to as the drop thing. The precondition for this action is that the agent must currently possess the specified thing. The effect of this action is that the agent will no longer possess the specified thing.

#### Tasks:

**Swap two things:** This compound task enables the AI to swap two objects. The AI will need to specify two objects, referred to as x thing and y thing. There are two methods to achieve this task. The first method is applicable when the agent has the first thing and does not have the second thing. In this case, the agent drops the first thing and then picks up the second thing. The second method is applicable when the agent has the second thing and does not have the first thing. In this case, the agent drops the second thing and then picks up the first thing.

## Output S2

### Output for S2 from GPT-4o for basic domain

```
(define (domain object-management)
  (:requirements :strips :typing)

  (:types
   thing
  )

  (:predicates
   (have ?t - thing)
  )

  (:action pickup
   :parameters (?t - thing)
   :precondition ()
   :effect (and (have ?t))
  )

  (:action drop
   :parameters (?t - thing)
   :precondition (have ?t)
   :effect (and (not (have ?t)))
  )

  (:task swap-things
   :parameters (?x - thing ?y - thing)
   (:method swap-method-1
    :precondition (and (have ?x) (not (have ?y)))
    :tasks (sequence (drop ?x) (pickup ?y))
   )
   (:method swap-method-2
    :precondition (and (have ?y) (not (have ?x)))
    :tasks (sequence (drop ?y) (pickup ?x))
   )
  )
)
```



## **Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature