LLMs as Planning Modelers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models

Marcus Tantakoun[1], **Xiaodan Zhu**[2], **Christian Muise**[3]

[1]School of Computing, Queen's University, Kingston, Canada
{20mt1, xiaodan.zhu, christian.muise}@queensu.ca

Abstract

Large Language Models (LLMs) excel in various natural language tasks but often struggle with long-horizon planning problems requiring structured reasoning. This limitation has drawn interest in integrating neuro-symbolic approaches within the Automated Planning (AP) and Natural Language Processing (NLP) communities. However, identifying optimal AP deployment frameworks can be daunting. This paper aims to provide a timely survey of the current research with an in-depth analysis, positioning LLMs as tools for extracting and refining planning models to support reliable AP planners. By systematically reviewing the current state of research, we highlight methodologies, and identify critical challenges and future directions, hoping to contribute to the joint research on NLP and Automated Planning.

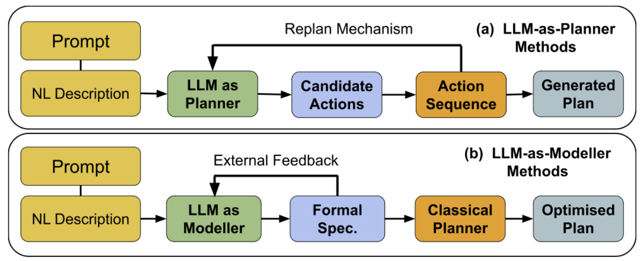

Figure 1: Distinction of LLM-planning: (a) uses LLMs for direct planning; (b) uses LLMs to generate AP specifications for existing task planning methods such as PDDL.

1 Introduction

Large Language Models (LLMs) has marked a significant paradigm shift in AI, sparking claims regarding emergent reasoning capabilities within LLMs (Wei et al. 2022a) and their potential integration into automated planning for agents (Pallagani et al. 2023). While LLMs, due to the prowess of distributed representation and learning, excel at System I tasks, planning—an essential aspect of System II cognition (Daniel 2017)—remains to be a significant bottleneck (Bengio 2020). Furthermore, LLMs face challenges with long-term planning and reasoning, often producing unreliable plans (Valmeekam, Stechly, and Kambhampati 2024; Pallagani et al. 2023; Momennejad et al. 2023), frequently failing to account for the effects and requirements of actions as they scale (Stechly, Valmeekam, and Kambhampati 2024), and their performance degrades with self-iterative feedback (Stechly, Marquez, and Kambhampati 2023; Valmeekam, Marquez, and Kambhampati 2023; Huang et al. 2024a).

Acknowledging the challenges of LLMs' *direct* planning capabilities, Automated Planning (AP), or AI planning, presents a promising alternative to generate robust, optimal plans through logical and computational methods. Meanwhile, LLMs excel at extracting and refining classical planning models from Natural Language (NL), leaving the plan generation to classical planners and heuristics.

This paper is driven by the fragmented landscape in the current literature, where many surveys lack a cohesive overview of LLM integration within this field. Our focus stems from the need to address these gaps and provide a clear framework, highlighting the importance of aligning LLM capabilities with areas where they offer tangible benefits. The motivation for this approach is threefold: (i) **Planning Accuracy**: LLMs can help ensure that all relevant factors are considered, reducing the risk of overlooked constraints (Huang, Lipovetzky, and Cohn 2024). (ii) **Adaptability**: LLMs can aid systems in adapting to dynamic environments to capture real-world nuances better, reducing the need for manual edits (Dagan, Keller, and Lascarides 2023). (iii) **Agnostic Modeling**: LLMs enable domain-agnostic models reducing reliance on specialized expertise, a major bottleneck in AP (Gestrin, Kuhlmann, and Seipp 2024).

Our survey's taxonomy is divided into three key areas: **Model Generation**, further consisting of (i) *Task Generation* (sec. 3.1.1), which translates initial and goal states along with object assignments; (ii) *Domain Generation* (sec. 3.1.2), constructing predicates and action schemas; and (iii) *Hybrid Generation* (sec. 3.1.3), encapsulating both task instance and domain generation. Model refinement and repair are addressed in **Model Editing** (sec. 3.2), while LLM-AP benchmarks are discussed in **Model Benchmarks** (sec. 3.3).

To our knowledge, this is the first comprehensive survey of LLM-driven AP-model construction. Our contributions can be summarized as follows:

- A critical survey of LLM-driven Automated Planning (AP) model generation, editing, and AP-LLM benchmarks, structured within our taxonomy.
- A summary of both shared and novel technical ap-

proaches for integrating LLMs into AI planning frameworks alongside their limitations.

- We provide insights on key challenges and opportunities, outlining future research directions for the community. To support future work, we provide **Language-to-Plan (L2P)**, an open-source Python library that implements landmark papers covered in this survey.

We hope this paper will contribute to and facilitate the joint research on Automated Planning and NLP.

2 Background

2.1 Automated Planning

Automated Planning (AP), focuses on synthesizing action sequences to transition from initial to goal states within its environmental constraints. Its most recognized category is the Classical Planning Problem (Russell and Norvig 2020). This can be formally defined as a tuple $\mathcal{M} = \langle \mathcal{S}, s^{\mathcal{I}}, s^{\mathcal{G}}, \mathcal{A}, \mathcal{T} \rangle$ where $\mathcal{S}$ is a finite and discrete set of states used to describe the world such that each state $s \in S$ is defined by the values of a fixed set of variables. $s^{\mathcal{I}} \in \mathcal{S}$, $s^{\mathcal{G}} \subseteq \mathcal{S}$ represent the initial state and goal world states, respectively. $\mathcal{A}$ is a set of symbolic actions, and $\mathcal{T}$ is the underlying transition function which takes the current state s^i and an action $a \subseteq \mathcal{A}$ as input and outputs the corresponding next state $\mathcal{T}(s^i, a) = s^{i+1}$. A solution to a planning problem $\mathcal{P}$ is a plan ϕ consists of a sequence of actions $\langle a_1, a_2, ..., a_n \rangle$ such that the preconditions of a_1 hold in $s^{\mathcal{I}}$, the preconditions of a_2 hold in the state that results from applying a_1, and so on, with the goal conditions all holding in the state that results after applying a_n.

> Consider a classic planning scenario where a delivery truck distributes packages. The state space $\mathcal{S}$ represents all package arrangements (e.g., in the warehouse, on the truck, or delivered). The initial state $s^{\mathcal{I}}$ places packages in warehouses, while the goal state $s^{\mathcal{G}}$ defines their destinations. The action set $\mathcal{A}$ includes "load," "unload," and "move," with transitions governed by $\mathcal{T}$. A valid plan ϕ might involve driving to a warehouse, loading packages, delivering them.

2.2 PDDL

As a fundamental building block of AP, the Planning Domain Definition Language (PDDL) (McDermott et al. 1998) is among the most used method of encoding planning tasks. The PDDL representation comprises of two files: domain $\mathbb{DF}$ and problem $\mathbb{PF}$. $\mathbb{DF}$ defines the universal aspects of a problem, highlighting the underlying fixed set of rules and constraints. This consists of predicates defining the state space $\mathcal{S}$ and set of actions $\mathcal{A}$. Each $a \subseteq \mathcal{A}$ is broken down into parameters *Par*(a) defining what types are being used in the action, the preconditions *Pre*(a), and subsequent effects *Eff*(a), encapsulating the transition function $\mathcal{T}$. $\mathbb{PF}$ consists of a list of objects that ground the domain, the problem's initial $s^{\mathcal{I}}$ and goal states $s^{\mathcal{G}}$. We provide a concrete example in Appendix A. The standardization and use of PDDL in planning have strongly facilitated sharing and benchmarking. This allows a wide selection of tools to support validation and refinement of models. Due to its flexibility, clear syntax, and declarative nature, it aligns well with LLMs' capabilities to translate descriptions into PDDL, as all modern LLMs should have encountered PDDL code in their training corpora (see Appendix A.1 for more details).

2.3 Large Language Models + Planning

Large Language Models (LLMs) have shown promise in generating highly structured outputs from natural language (NL) descriptions. Frameworks like StarCoder (Li et al. 2023), CodeGen (Nijkamp et al. 2023) and PLANSEARCH (Wang et al. 2024) have demonstrated LLMs' ability to translate NL to executable code. Previous work has translated formal intermediate representations such as Temporal Logic (Chen et al. 2023b; Pan, Chou, and Berenson 2023; Mao et al. 2024; Cosler et al. 2023; Liu et al. 2023b; Luo, Xu, and Liu 2023; Fuggitti and Chakraborti 2023), Answer Set Programming (ASP) (Yang, Ishay, and Lee 2023), and other variations of the paradigm (Pan et al. 2023; Wang et al. 2023; Chen et al. 2023a). Preliminary PDDL-LLM works are found in (Miglani and Yorke-Smith 2020; Feng, Zhuo, and Kambhampati 2018; Simon and Muise 2021; Chalvatzaki et al. 2023). Currently, researchers are further exploring the nuances of varying pipelines to balance the effectiveness and limitations of LLMs in building such neuro-symbolic frameworks. Huang et al. (2024b) survey a limited amount of works to compose their high-level abstraction of LLM-augmented planning agents. Pallagani et al. (2024) go beyond the scope of traditional AP, encapsulating more broader constructs, whereas Zhao et al. (2024) provide an extensive overview of LLM-TAMP applications. Most akin to our paper, Li et al. (2024a) and Wei et al. (2025) review studies using LLMs in planning with PDDL; however, their survey mainly consists of frameworks comprised of *LLMs-as-Planners*, rather than *LLM-as-Modelers* (cf. Figure 1).

> **Scope of Survey**: Broadly, LLMs+AP paradigms can be categorized as: (i) *LLMs-as-Heuristics*, where they enhance search efficiency by providing heuristic guidance (Silver et al. 2022; Tuisov, Vernik, and Shleyfman 2025; Sel, Jia, and Jin 2025); (ii) *LLMs-as-Planners*, either directly involved with the decision of action sequences (Zhang et al. 2024c) or proposing plans that are later refined through post-hoc implementations (Gundawar et al. 2024; Arora and Kambhampati 2023; Pallagani et al. 2022; Burns, Hughes, and Sycara 2024); and (iii) *LLMs-as-Modelers*, where LLMs assist in constructing planners (Katz et al. 2024; Cao et al. 2024) or planning models—defining planning domains, extracting task representations, and refining problem formulations. Our paper surveys approximately 80 papers and technical reports of the latter, focusing on key research questions and potential directions for future work in the context of LLMs and planning models.

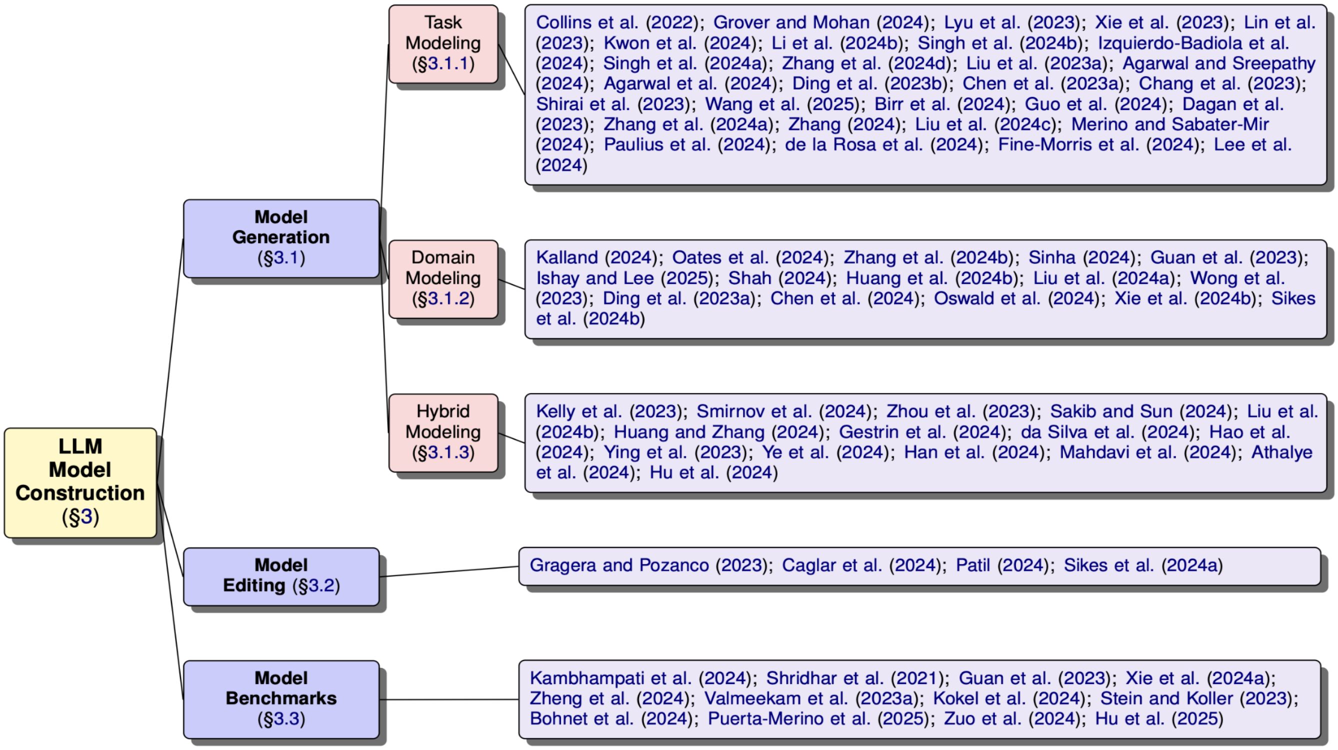

Figure 2: Taxonomy of research in LLM Model Construction

3 Large Language Models for Constructing Planning Models

We consider the research on leveraging powerful LLMs to assist in constructing planning models to be of critical importance and a more promising approach than relying on LLMs to perform planning directly (Vyas et al. 2025). Verifiable planning modules remain the backbone of planning, ensuring reliability, robustness, and explainability—core requirements for many planning tasks. Note that deploying LLMs themselves in an end-to-end manner to perform planning still falls short of providing soundness guarantees (Valmeekam et al. 2024) and inherently struggle with correctness and completeness (Katz et al. 2024). We organize the existing works into a taxonomy comprising three key areas: **Model Generation**, **Model Editing**, and **Model Benchmarks**, as shown in Figure 2. These tasks need joint efforts from the NLP and automated planning community.

3.1 Model Generation

This first group of works are centered around **Model Generation**—extracting components or generating complete planning models from the user or environmental natural language input. This is further divided into three aspects: *Task Modeling* (sec. 3.1.1), which translates objectives into initial conditions and goal states; *Domain Modeling* (sec. 3.1.2), which defines the foundational components like entities, actions, and relationships in the system; and *Hybrid Modeling* (sec. 3.1.3), which integrates both aspects to create a complete model, enabling end-to-end planning.

To facilitate further discussion on LLM-driven model construction, we highlight two key research questions:

RQ1 How can LLMs accurately align with human goals, ensuring these models correctly represent desired specifications and objectives?

RQ2 To what extent and granularity of detail can human-level instructions be effectively translated into accurate model definitions?

3.1.1 Task Modeling

For *goal-only specification*, Collins et al. (2022) and Grover and Mohan (2024) utilize few-shot prompting whereas **Faithful CoT** (Lyu et al. 2023) puts heavy emphasis on an interleaving technique of chain-of-thought (CoT) prompting (Wei et al. 2022b). Xie et al. (2023) assess the effectiveness of LLMs in translating tasks with varying levels of ambiguity in both NL and other languages such as Python. **Text2Motion** (Lin et al. 2023) leverage LLMs to generate goal predictions that are checked against a learned library of Q-function skills—based on feasibility—that support geometric feasibility in robotic tasks. Kwon, Kim, and Kim (2024) decomposes long-term tasks into sub-goals using LLMs, then executing task planning for each sub-goal with either symbolic methods or MCTS-based LLM planners. **Safe Planner** (Li et al. 2024b) uses an LLM to convert NL instructions into PDDL goals, enabling a closed-loop VLM-planner to operate based on real-time environmental observations. Fine-Morris et al. (2024) propose a framework that decomposes NL goals into predicate-based specifications, structured as Python dictionaries, which are then formatted into HDDL decomposition methods.

Branching to *multi-agent goal collaboration*, **DaTAPlan** (Singh et al. 2024) employs an LLM to predict high-level anticipated tasks against human actions, triggering re-planning or new task predictions when deviations occur. **PlanCollabNL** (Izquierdo-Badiola et al. 2024) allocates sub-goals among LLM agents which are then encoded into PDDL goals, and modifying the action costs based on LLM recommendations. **TwoStep** (Singh, Traum, and Thomason 2024) decomposes multi-agent planning problems into two single-agent problems with mechanisms to ensure smooth coordination. **LaMMA-P** (Zhang et al. 2024d) allocate sub-tasks composed of general action sequences, and generates PDDL problem descriptions for each robot's domain.

Frameworks that handle *complete* PDDL task specifications can be broadly categorized into open-loop and closed-loop approaches. For the former, **LLM+P** (Liu et al. 2023a) uses in-context examples of problems within the same domain to generate whole problem files. **TIC** (Agarwal and Sreepathy 2024) achieved nearly 100% accuracy with GPT-3.5 Turbo across LLM+P planning domains by translating the task into intermediate representations, refining them, and processing them through a logical reasoner. (Lee et al. 2024) address a critical real-world application by introducing **PlanAID**, a system designed to generate emergency operation plans (EOPs) by leveraging Retrieval-Augmented Generation (RAG) documents to enhance the interaction between the user and the LLM. For real-world grounding, recent work converts NL instructions into structured geometric representations that bridge abstract language understanding and spatial reasoning (Ding et al. 2023b; Chen et al. 2023a; Chang et al. 2023), while other approaches integrate LLMs with Vision Language Models (VLMs) to further ground language understanding in spatial contexts (Shirai et al. 2023; Wang et al. 2025).

For closed-loop approaches, **Auto-GPT+P** (Birr et al. 2024) generates the initial state of the problem based on visual perception and an automated error self-correction loop for the generated PDDL goal. Guo, Kingston, and Kavraki (2024) decompose the problem into both PDDL and Python specifications, incorporating a set of constraints into an SMT-based TAMP solver via a Python API. **LLM+DP** (Dagan, Keller, and Lascarides 2023) holds beliefs in uncertain environments to construct possible world states, dynamically updating its internal state and re-plans. **PDDLEGO** (Zhang et al. 2024a; Zhang 2024) performs a recursive task decomposition into sub-goals that enable the agent to gather new observations, progressively refining the problem file until it can develop a solvable plan. Liu et al. (2024c) integrate user information into a hierarchical scene graph of the environment, enabling an LLM to predict human activities and goal states, which are then refined using predicates and domain knowledge to ground problem specifications. Merino and Sabater-Mir (2024) model NPC behavior by leveraging LLMs conditioned on "memories" that represent environmental context.

Instead of extracting PDDL models directly, Paulius et al. (2024) introduce a modularized approach that leverages LLMs to produce Object-Level Plans (OLP), which describe high-level changes to object states and uses them to bootstrap TAMP hierarchically. **TRIP-PAL** (de la Rosa et al. 2024) translates intermediate representations via travel points of interest (POI) and user information into dictionaries, which are then converted into PDDL planning tasks.

Summary: Some methods directly translate natural language task descriptions into PDDL (Kelly et al. 2023; Liu et al. 2023a). Others enhance goal specification by incorporating reasoning chains and few-shot examples (Lyu et al. 2023), while (Dagan, Keller, and Lascarides 2023; Zhang et al. 2024a) target dynamic, online-planning applications. However, these current approaches rely on an explicit mapping between NL and PDDL code, requiring users to specify each state predicate in detail. To address ambiguity in minimal task descriptions, **future research** should develop methods capable of inferring complete and robust PDDL specifications from sparse input, building on prior work that has explored this concept via external perceptual groundings (Shirai et al. 2023), RAG implementations (Lee et al. 2024), or leveraging LLM commonsense capabilities to capture underlying assumptions and constraints (Agarwal and Sreepathy 2024).

3.1.2 Domain Modeling

Various works have executed domain modeling in a one-shot manner. Kalland (2024) combines a language and an automatic speech recognition model to generate PDDL specifications. To better understand cyber-attacks in real-time, **CLLaMP** (Oates et al. 2024) leverages LLMs to extract PDDL action models from Common Vulnerabilities and Exposures descriptions, finding that in-context (IC) examples are superior to CoT prompting. Zhang et al. (2024b) introduce **PROC2PDDL**, which proposes a *Zone of Proximal Development* prompt design—a variant of CoT (Vygotsky 1978). To mitigate misalignments with users requirements, Huang, Lipovetzky, and Cohn (2024) use multiple LLMs to generate candidate PDDL action schemas, which are then passed through a sentence encoder to compute the semantic relatedness of code and original NL descriptions.

Guan et al. (2023) recognize the impracticality of LLMs generating fully functioning PDDL models in a single query (Kambhampati et al. 2024). Their framework **LLM+DM** outlines a generate-test-critique approach (Romera-Paredes et al. 2024; Trinh et al. 2024), leveraging multiple LLM calls to incrementally build key components of the domain by a dynamically generated predicate list. Similarly, Ishay and Lee (2025) introduce **LLM+AL**, which uses LLMs to generate action languages, but instead, in BC+ syntax (extension of Answer Set Programming). Shah (2024) presents **LAMP**, an extensive series of proposed algorithms that learn abstract

PDDL domain models. Sinha (2024) proposes a structured prompt engineering approach to generate domain models in the Hierarchical Planning Definition Language (HPDL). Sikes et al. (2024b) explored decomposing NL information from operational models via Javascript functions into PDDL actions. **BLADE** (Liu et al. 2024a) bridges language-annotated human demonstrations and primitive action interfaces by tasking an LLM to define PDDL preconditions and effects conditioned on behaviors containing all possible sequences of contact primitive.

In terms of closed-loop frameworks, **ADA (Action Domain Acquisition)** (Wong et al. 2023) tasks LLMs with generating candidate symbolic task decompositions, extracting undefined action names, and iteratively prompting for their definitions. **COWP** (Ding et al. 2023a) handles unforeseen situations in open-world planning by storing the robot's closed-world state when planning fails, triggering a "Knowledge Acquirer" module that leverages LLMs to augment action preconditions and effects. Unlike COWP, which relies on predefined error factors, **LASP** (Chen et al. 2024) identifies potential errors from environmental observations, using an LLM to generate error causes in NL, suggesting action preconditions. Xie et al. (2024b) use fine-tuned LLMs for precondition and effect inference from NL actions, and semantic matching to validate actions by comparing inferred preconditions with the current world states.

Oswald et al. (2024) shifts focus to assessing domain quality, addressing limitations of manual human evaluation (Hayton et al. 2020; Huang, Chen, and Zhang 2014). This study measures equivalence to the ground truth in terms of *operational equivalence*—whether reconstructed domains behave identically to the original by agreeing on the validity of action sequences as plans. To achieve this, the authors decompose ground truth PDDL actions into NL using an LLM, which then tasks them again to reconstruct PDDL domain models for quality assessment.

Summary: Kambhampati et al. (2024); Wong et al. (2023) use incremental methods that iteratively generate and refine models, while incorporating real-world examples have shown to enhance contextual input accuracy (Oates et al. 2024; Ding et al. 2023a). The complexity of these frameworks demonstrate that constructing domains are inherently more challenging than task specification. However, current methods generate only a single domain, which risks misalignment with human expectations by failing to capture all necessary constraints. **Future research** should explore methods that generate multiple candidate domains (Huang, Lipovetzky, and Cohn 2024) and incorporate iterative refinement mechanisms to better align with user requirements.

3.1.3 Hybrid Modeling

Kelly et al. (2023) extract narrative planning domains and problems from input stories using a one-shot prompt, iterating with a second prompt conditioned on the planner's error message until a successful plan is found. Smirnov et al. (2024) utilize pre-processing steps like JSON markup generation, consistency checks, and error correction loops. Their framework also includes a "reachability analysis" pipeline to extract feedback from flawed domains or unreachable problems, alongside a dependency analysis to check predicate usage across both files. **ISR-LLM** (Zhou et al. 2023) does not offer any feedback mechanisms to fix PDDL specifications; however, it does introduce self-refinement during the plan generation phase by incorporating the external validator tool, VAL (Howey, Long, and Fox 2004). Sakib and Sun (2024) generate multiple high-level task plans in Knowledge Graphs (KG), refine the network by removing unreliable components, and feed the task plan to an LLM to extract the PDDL domain and problem files for low-level robot skills. Conversely, **DELTA** (Liu et al. 2024b) initially generates PDDL files and a scene graph, followed by pruning unnecessary details from the graph to focus on relevant items.

Huang and Zhang (2024) further support that LLMs are prone to one-shot generation errors, highlighting the need for intermediate representations before converting to PDDL. **NL2Plan** (Gestrin, Kuhlmann, and Seipp 2024) is the first domain-agnostic offline end-to-end NL planning system, requiring only minimal description and using pre-processing and automated common sense feedback to interface between the LLM and the user. **LLM4CAP** (da Silva et al. 2024) reduces manual effort, with an LLM-generated ontology being iteratively verified using an LLM to check for syntax errors, hallucinations, and missing elements. **LLMFP** (Hao, Zhang, and Fan 2024) translates goals, decision variables, and constraints into a JSON representation, which is then used to generate Python code for an SMT solver to produce plans without task-specific examples or external critics. **NIPE** (Ying et al. 2023) leverages LLMs as few-shot semantic parsers to generate conditional statements from spatial descriptions, guiding PDDL sampling and action model definition for Bayesian goal inference.

Previous works tackle open-world settings, with **MO-PRHeus** (Ye et al. 2024) focusing on human-in-the-loop long-horizon planning, introducing an anomaly detection mechanism to identify potential execution errors and update corresponding PDDL files to reflect changes in the world model. **InterPret** (Han et al. 2024) uses LLMs to enable robots to learn PDDL predicates and derive action schemas through interactive language feedback from non-expert users via Python perception APIs. Like other frameworks, Mahdavi et al. (2024) uses environmental interactions for evaluation and verification, starting with the LLM defining candidate PDDL problem files and domain sets, which are then refined through iterative cycles using their novel Exploration Walk (EW) method. Athalye et al. (2024) introduces **pix2pred**, a framework that leverages VLMs to propose predicates and determine their truth values in demonstrations. By bridging raw sensor data with symbolic planning, it structures the invented predicates to generalize across complex, long-horizon tasks, enabling the conversion of image data into planner-compatible representations, such as those based on PDDL.

Instead of generating planning models to be used by external planners, **AgentGen** (Hu et al. 2024) uses LLMs to synthesize diverse environments and planning tasks (in languages such as PDDL) and their corresponding NL descriptions, targeted for LLM-based agent training. The authors found that LLMs instruction tuned with this dataset achieved significant improvement in both in-domain and out-of-domain planning tasks. As an exploration, (Hu et al. 2025) uses this dataset to run supervised fine-tuning with the *LLaMA-3.1* models and found significant improvements in generating domain models–where larger models tend to benefit more from supervised fine-tuning.

Summary: Complexities arise due to the need to coordinate between the domain and the problem. Human-in-the-loop interactions are frequently employed (Kelly et al. 2023), whereas other methods incorporate pre-processing steps–involving external tools like FastDownward and VAL (Zhou et al. 2023; Smirnov et al. 2024) or custom-designed rules (Mahdavi et al. 2024; Gestrin, Kuhlmann, and Seipp 2024). Yet, these linear pipelines risk cascading errors–such as the possibility of new objects in the task, prompting new PDDL types in the domain. **Future work** should focus on modularity, such as enabling dynamic integration of types and predicates in later stages of generation. This would result in more adaptable and error-tolerant planning systems.

3.2 Model Editing

Model edits have gained traction as a promising application functioning more as assistive tools than fully autonomous generative solutions. Insights into the likelihood of various model outcomes can support authors in refining models with greater efficiency toward an automated approach.

Gragera and Pozanco (2023) investigate the limitations of LLMs in repairing unsolvable tasks caused by incorrect task specifications, assessing the effectiveness of prompting in both PDDL and NL. As derivative work, Caglar et al. (2024) tackle the challenge of modifying model spaces beyond classical planning by exploring the effectiveness of LLMs in generating more plausible model edits, particularly concerning unsolvability and plan executability, either as a direct replacement or in augmentation of combinatorial search (CS) methods and manual authoring. Patil (2024) conduct a comprehensive study on using LLMs, with traditional error-checking methods, to detect and correct syntactic and semantic errors in PDDL domains, demonstrating that while LLMs excel at identifying and fixing syntactic issues, their performance in resolving semantic inconsistencies is less reliable. Sikes et al. (2024a) address the issues of planning models with domains containing a set of semantically equivalent state variables, which are syntactically different–provided with different labels. Specifically, they address failures in scenarios where information about the model is coming from heterogeneous sources.

Summary: Current research shows promise in leveraging LLMs to correct syntactic errors, but addressing semantic errors remains a significant challenge (Patil 2024). To overcome these issues, **future work** should explore post-hoc correction strategies. For instance, researchers could develop methodologies to analyze plan outputs after generation, identifying semantic gaps or inconsistencies through automated metrics or even human-in-the-loop evaluation systems. Such feedback can iteratively refine planning models, ensuring plans are both syntactically correct and semantically faithful.

3.3 Model Benchmarks

Non-deterministic output behaviors inherent in LLMs make it challenging to assess the quality of frameworks used in planning benchmarks. This heightens the importance of robustness, especially for evaluating LLMs' ability to extract planning models (Behnke and Bercher 2024). However, this also introduces challenges of automating evaluation with these extracted models, determining levels of ambiguity in natural language inputs and assumptions, and monitoring fine-granular modes of failure in generation.

PlanBench (Valmeekam et al. 2023) aims to systematically evaluate LLM planning capabilities with an emphasis on cost-optimal planning and plan verification. **ACPBench** (Kokel et al. 2024) standardizes evaluation tasks and metrics for assessing reasoning about actions, changes (transitions), and planning—across 13 domains, on 22 SOTA language models including OpenAI's o1 reasoning model. **AutoPlanBench** (Stein and Koller 2023) first converts PDDL planning benchmarks into NL via LLMs, and then tasks LLMs to produce a plan through various prompting techniques. Bohnet et al. (2024) provide a scalable benchmark suite in both PDDL and NL to measure the planning capabilities of LLMs in various strategies, as well as providing a mapping method for translating PDDL benchmarks to NL and measuring the performance of the generated benchmarks. Puerta-Merino et al. (2025) propose a road map and benchmark to address the gap of LLM integration in Hierarchical Planning (HP).

To determine whether testing PDDL domains have been leaked to training data of LLMs, **Mystery Blocksworld** (Kambhampati et al. 2024) obfuscates the classic **Blocksworld** (Gupta, Efros, and Hebert 2010) planning problem by altering the named types to be syntactically nonsensical. **ALFWorld** and **Household** (Shridhar et al. 2021; Guan et al. 2023) tackles the complexities of real-world typical household environment that uses PDDL semantics to produce textual observations. **TravelPlanner** (Xie et al. 2024a) assesses language models' abilities in planning through agent-based interactions in a travel-planning environment. Zheng et al. (2024) extend this work with **Natural Plan**, which evaluates LLMs on realistic planning and scheduling benchmarks using APIs.

Previous LLM+AP benchmarks only focus on LLM plan generation capabilities, whereas **Planetarium** (Zuo et al. 2024) rigorously evaluates LLM-generated PDDL task specifications, highlighting two issues: (i) LLMs can produce valid code that misaligns with the original NL description, and (ii) evaluation sets often use NL descriptions too similar to the ground truth, reducing the task's challenge. The benchmark assesses LLMs' ability to generate PDDL problems across varying levels of abstraction and size. On the other hand, **Text2World** (Hu et al. 2025) introduces an automated pipeline for domain extraction and rigorous multi-criteria metrics that address the limitations of narrow domain scope (Liu et al. 2023a) and indirect evaluation methods in end-to-end plan assessments (Guan et al. 2023). Key metrics, including executability, structural similarity, and component-wise F1 scores, are employed while exploring state-of-the-art LLMs and fine-tuning techniques for generating and evaluating PDDL models.

Summary: Assessing the quality of LLM-generated PDDL models (Zuo et al. 2024; Oswald et al. 2024) has made significant progress toward rigorous evaluation; however, the rapid leakage of training data to LLMs remains a major challenge, with (Hu et al. 2025) reporting high contamination rates in the evaluation domains from (Guan et al. 2023). "Mystifying" the domains (Kambhampati et al. 2024) does not address LLMs' limitations in handling real-world modeling tasks. Other concerns arise that even with "mystified" domains, LLMs may still memorize code structures, rendering string obfuscation ineffective (Chen et al. 2025). **Future work** should explore solutions for routinely changing benchmark standards for domains, actively involving the planning community in its ongoing refinement. Khandelwal, Sheth, and Agostinelli (2024) proposes a tool for generating diverse and complex planning domains, which could serve as a foundation for such a benchmark.

4 (L2P) Library

With the proliferation of related techniques to convert NL to PDDL, we are seeing an ever-increasing set of related methods. To bring them together under a single computational umbrella, and beyond just relating the work together conceptually as we have done thus far in this survey, we created a unified platform: **L2P** [1], which re-implements landmark papers covered in this survey. This Python library is open source and has the capability of encapsulating the generalized version of the proposed "LLM-Modulo" framework (Kambhampati et al. 2024), which emphasizes soundness guarantees through iterative plan refining via external verifiers. While L2P embodies the core principles of the LLM-Modulo framework, which advocates for LLMs autonomously generating plan candidates themselves, it shifts the focus by facilitating the creation of PDDL files using LLMs – aligning with this paper's paradigm. This approach allows for the integration of external verifiers and feedback critics, enabling users to modify and refine the extracted models. L2P offers three major benefits:

(i) **Comprehensive Tool Suite**: users can easily plug in various LLMs for streamlined extraction experiments with our extensive collection of PDDL extraction and refining tools.

(ii) **Modular Design**: facilitates flexible PDDL generation, allowing users to explore prompting styles and create customized pipelines.

(iii) **Autonomous Capability**: supports a fully autonomous pipeline, reducing the need for manual authoring.

Appendix B demonstrates several usage examples of L2P. To demonstrate the flexibility of the framework, we re-implemented several of the landmark papers covered in this survey, and Appendix C provides potential works that can be recreated by L2P at the time of writing. We hope that the community contributes to the L2P framework as a repository of existing advancements in LLM model acquisition and relevant papers, ensuring that users have access to the most current research and tools under a common framework for fair comparison.

4.1 Paper Reconstructions

L2P can recreate and encompass previous frameworks for converting natural language to PDDL, serving as a comprehensive foundation that integrates past approaches. The L2P GitHub contains multiple paper reconstructions as examples of how existing methods can be implemented and compared within a unified system, demonstrating L2P's flexibility and effectiveness in standardizing diverse NL-to-PDDL techniques. An example of Guan et al. (2023) "action-by-action" algorithm can be found in Figure 3; predicate output can be found in Appendix B Figure 7.

5 Discussion

Revisiting RQ1: While frameworks can generate parsable and solvable PDDL files, it remains uncertain if these specifications align with human goals. Simple domains like Blocksworld are easier to verify. Still, scaling complex domains require users to understand how LLMs generate these specifications, emphasizing the need for explainable planning for robust, transparent, and correctable outputs (Zuo et al. 2024). Corrective feedback loops have demonstrated significant improvements in resolving failing actions preconditions (Raman et al. 2024), or re-planning in case of unexpected failures during the execution of the plan (Joublin et al. 2023; Raman et al. 2022). Ensuring alignment with user goals involves breaking down PDDL model construction into pre-processing steps with human-in-the-loop feedback, as seen in frameworks like (Kelly et al. 2023). Very reminiscent of the "critics" process in the LLM-Modulo framework (Kambhampati et al. 2024), setting up a sort of external verifier checklist and using LLMs to provide feedback, is demonstrated by Gestrin, Kuhlmann, and Seipp (2024) and Smirnov et al. (2024). A strong idea is analyzing the semantic correctness of plans generated and using that as feedback to refine the LLM-generated PDDL specifications (Sakib and Sun 2024). Additionally, intermediate representation (i.e. ASP, Python, JSON) methods that are easier for LLMs to process before converting to PDDL (Agarwal

[1]Code made publicly at: https://github.com/AI-Planning/l2p

```
import os
from l2p import *

def run_aba_alg(model: LLM, action_model,
    domain_desc, hierarchy, prompt, max_iter: int=2
    ) -> tuple[list[Predicate], list[Action]]:
actions = list(action_model.keys())
pred_list = []

for _ in range(max_iter):
  action_list = []
  # iterate each action spec. + new predicates
  for _, action in enumerate(actions):
    if len(pred_list) == 0:
      prompt = prompt.replace('{predicates}',
      '\nNo predicate has been defined yet')
    else: res = ""
      for i, p in enumerate(pred_list):
        res += f'\n{i + 1}. {p["raw"]}'
        prompt = prompt.replace('{predicates}', res)
    # extract pddl action and predicates (L2P)
    pddl_action, new_preds, response = (
      builder.extract_pddl_action(
        model=model,
        domain_desc=domain_desc,
        prompt_template=prompt,
        action_name=action,
        action_desc=action_model[action]['desc'],
        action_list=action_list,
        predicates=pred_list,
        types=hierarchy["hierarchy"]
      )
    )
    new_preds = parse_new_predicates(response)
    pred_list.extend(new_preds)
    action_list.append(pddl_action)
  pred_list = prune_predicates(pred_list,action_list)
return pred_list, action_list

if __name__ == "__main__":
  builder = DomainBuilder() # domain build class
  # load in assumption files
  base_path='paper_reconstructions/llm+dm/prompts/'
  action_model=load_file(
      f'{base_path}action_model.json')
  domain_desc=load_file(
      f'{base_path}domain_desc.txt')
  hier=load_file(
      f'{base_path}hierarchy_requirements.json')
  prompt=load_file(f'{base_path}pddl_prompt.txt')
  # initialise LLM engine (OpenAI in this case)
  api_key = os.environ.get('OPENAI_API_KEY')
  llm = OPENAI(model="gpt-4o-mini", api_key=api_key)

  # run "action-by-action" algorithm
  pred, action = run_aba_alg(model=llm, action_model,
      domain_desc, hier, prompt)
```

Figure 3: Shortened L2P reconstruction of "action-by-action algorithm" (Guan et al. 2023). This iterates through each action description and generates its PDDL specifications while maintaining a dynamically generating predicate list.

and Sreepathy 2024; Smirnov et al. 2024) can also enhance accuracy. However, over time, LLMs are likely to improve, with fewer syntax errors and more logical semantic outputs as they train on more PDDL knowledge.

Revisiting RQ2: LLMs have demonstrated that they are significantly sensitive to prompting raising questions about whether they are better off functioning as machine *translators* or *generators*. Liu et al. (2023a) demonstrate that highly explicit descriptions improve translation accuracy, while Gestrin, Kuhlmann, and Seipp (2024) and Smirnov et al. (2024) leverage minimal descriptions, relying on LLMs' internal world knowledge to enrich outputs. Huang, Lipovetzky, and Cohn (2024), Liu et al. (2023a), and Guan et al. (2023) recognize that specifying a precise predicate set in natural language is crucial and addresses the common problem of evaluating across different methods. The challenge of operating with minimal to no textual guidance beyond the initial task prompt becomes significantly more complex in online planning contexts. This underscores the importance of standardizing prompt granularity for initial generation and iterative feedback (Liu et al. 2024b). Nabizada et al. (2024) provides a promising organized and standardized paradigm of automated generated PDDL descriptions that can be applied to LLMs.

5.1 Literature Scope and Additional Outlooks

Our survey examines nearly 80 works, with over 50 integrating LLMs into the workflow for generating AP specifications. These frameworks span task (30 papers), domain (15 papers), and hybrid (15 papers) model generation. Strong emphasis on task modeling stems from the fact that many systems, already operate within well-defined domains–such as robotics. However, advancing toward more sophisticated and adaptable systems–such as dialogue agents–requires a more holistic approach, where both task objectives and domain constraints are dynamically constructed. We encourage researchers to use this survey as a foundation for exploring beyond task modeling and toward integrated domain and hybrid specification approaches, enabling more robust and scalable end-to-end planning.

6 Conclusion

The issue of extracting these planning models has long been recognized as a major barrier to the widespread adoption of planning technologies, a topic that has been explored in depth within the knowledge acquisition literature in AP for decades (Vallati and Kitchin 2020; Hendler, Tate, and Drummond 1990). Even with the emergence of LLMs, this remains a persistent challenge, introducing a new suite of obstacles. In this survey, we examine nearly 80 scholarly articles that propose their frameworks and some other subsidiary works delegating model acquisition tasks to LLMs. By identifying the research distribution and gaps within these categories, we aim to provide a higher generalization from each framework's methodologies into broader aspects for future architectures. Additionally, we hope researchers can apply these methodologies to more advanced planning languages with the support of our L2P library–that can be found here: https://github.com/AI-Planning/l2p.

Limitations

This survey has two primary limitations. First, regarding its scope, our focus is limited to PDDL construction frameworks and related papers. Techniques remain largely unexplored in this context, and LLM capabilities in planning are still in their early stages. Due to page space constraints, we provide only a brief overview of each work rather than an exhaustive technical analysis. Additionally, our study primarily draws works published in ACL, ACM, AAAI, NeurIPS, ICAPS, COLING, CoRR, ICML, ICRA, EMNLP, and arXiv, so there is a possibility that we may have missed relevant research from other venues. Secondly, our L2P library currently supports only basic PDDL extraction tools for fully observable deterministic planning, and does not yet include tools for areas such as conditional, temporal, or numeric planning. We plan to expand the library to cover a broader range of PDDL applications, aiming to further research into the challenges LLMs encounter in these areas.

Ethics Statement

This survey does not pose any ethical issues beyond those already present in the existing literature on planning model construction via LLMs. As with any sufficiently advanced technology, there is an opportunity for misuse of the proposed L2P library (e.g., extracting actionable planning models for unethical domains). However, we view this as a pervasive issue with all of the existing methods that aim to extract planning theories from natural language.

References

Agarwal, S.; and Sreepathy, A. 2024. TIC: Translate-Infer-Compile for accurate 'text to plan' using LLMs and logical intermediate representations. *CoRR*, abs/2402.06608.

Arora, D.; and Kambhampati, S. 2023. Learning and Leveraging Verifiers to Improve Planning Capabilities of Pretrained Language Models. *CoRR*, abs/2305.17077.

Athalye, A.; Kumar, N.; Silver, T.; Liang, Y.; Lozano-Pérez, T.; and Kaelbling, L. P. 2024. Predicate Invention from Pixels via Pretrained Vision-Language Models. *arXiv preprint arXiv:2501.00296*.

Behnke, G.; and Bercher, P. 2024. Envisioning a Domain Learning Track for the IPC.

Bengio, Y. 2020. Deep Learning for System 2 Processing. *AAAI 2020*.

Birr, T.; Pohl, C.; Younes, A.; and Asfour, T. 2024. AutoGPT+P: Affordance-based Task Planning using Large Language Models. In *Robotics: Science and Systems XX*, RSS2024. Robotics: Science and Systems Foundation.

Bohnet, B.; Nova, A.; Parisi, A. T.; Swersky, K.; Goshvadi, K.; Dai, H.; Schuurmans, D.; Fiedel, N.; and Sedghi, H. 2024. Exploring and Benchmarking the Planning Capabilities of Large Language Models. *CoRR*, abs/2406.13094.

Burns, O.; Hughes, D.; and Sycara, K. P. 2024. PlanCritic: Formal Planning with Human Feedback. *CoRR*, abs/2412.00300.

Caglar, T.; Belhaj, S.; Chakraborti, T.; Katz, M.; and Sreedharan, S. 2024. Can LLMs Fix Issues with Reasoning Models? Towards More Likely Models for AI Planning. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, 20061–20069. AAAI Press.

Cao, D.; Katz, M.; Kokel, H.; Srinivas, K.; and Sohrabi, S. 2024. Automating Thought of Search: A Journey Towards Soundness and Completeness. *CoRR*, abs/2408.11326.

Chalvatzaki, G.; Younes, A.; Nandha, D.; Le, A. T.; Ribeiro, L. F. R.; and Gurevych, I. 2023. Learning to Reason over Scene Graphs: A Case Study of Finetuning GPT-2 into a Robot Language Model for Grounded Task Planning. *CoRR*, abs/2305.07716.

Chang, H.; Gao, K.; Boyalakuntla, K.; Lee, A.; Huang, B.; Kumar, H. U.; Yu, J.; and Boularias, A. 2023. LGMCTS: Language-Guided Monte-Carlo Tree Search for Executable Semantic Object Rearrangement. *CoRR*, abs/2309.15821.

Chen, G.; Yang, L.; Jia, R.; Hu, Z.; Chen, Y.; Zhang, W.; Wang, W.; and Pan, J. 2024. Language-Augmented Symbolic Planner for Open-World Task Planning. *CoRR*, abs/2407.09792.

Chen, W.; Zhang, L.; Zhong, L.; Peng, L.; Wang, Z.; and Shang, J. 2025. Memorize or Generalize? Evaluating LLM Code Generation with Evolved Questions.

Chen, Y.; Arkin, J.; Zhang, Y.; Roy, N.; and Fan, C. 2023a. AutoTAMP: Autoregressive Task and Motion Planning with LLMs as Translators and Checkers. *CoRR*, abs/2306.06531.

Chen, Y.; Gandhi, R.; Zhang, Y.; and Fan, C. 2023b. NL2TL: Transforming Natural Languages to Temporal Logics using Large Language Models. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, 15880–15903. Association for Computational Linguistics.

Collins, K. M.; Wong, C.; Feng, J.; Wei, M.; and Tenenbaum, J. B. 2022. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. *arXiv preprint arXiv:2205.05718*.

Cosler, M.; Hahn, C.; Mendoza, D.; Schmitt, F.; and Trippel, C. 2023. nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models. *CoRR*, abs/2303.04864.

da Silva, L. M. V.; Köcher, A.; Gehlhoff, F.; and Fay, A. 2024. Toward a Method to Generate Capability Ontologies from Natural Language Descriptions. *CoRR*, abs/2406.07962.

Dagan, G.; Keller, F.; and Lascarides, A. 2023. Dynamic Planning with a LLM. *CoRR*, abs/2308.06391.

Daniel, K. 2017. *Thinking, fast and slow*. Macmillan.

de la Rosa, T.; Gopalakrishnan, S.; Pozanco, A.; Zeng, Z.; and Borrajo, D. 2024. TRIP-PAL: Travel Planning with Guarantees by Combining Large Language Models and Automated Planners. *CoRR*, abs/2406.10196.

Ding, Y.; Zhang, X.; Amiri, S.; Cao, N.; Yang, H.; Kaminski, A.; Esselink, C.; and Zhang, S. 2023a. Integrating Action Knowledge and LLMs for Task Planning and Situation Handling in Open Worlds. *CoRR*, abs/2305.17590.

Ding, Y.; Zhang, X.; Paxton, C.; and Zhang, S. 2023b. Task and Motion Planning with Large Language Models for Object Rearrangement. *CoRR*, abs/2303.06247.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. *CoRR*, abs/1803.02632.

Fine-Morris, M.; Hsiao, V.; Smith, L. N.; Hiatt, L. M.; and Roberts, M. 2024. Leveraging LLMs for Generating Document-Informed Hierarchical Planning Models: A Proposal. In *AAAI 2025 Workshop LM4Plan*.

Fuggitti, F.; and Chakraborti, T. 2023. NL2LTL - a Python Package for Converting Natural Language (NL) Instructions to Linear Temporal Logic (LTL) Formulas. In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 16428–16430. AAAI Press.

Gestrin, E.; Kuhlmann, M.; and Seipp, J. 2024. NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions. *CoRR*, abs/2405.04215.

Gragera, A.; and Pozanco, A. 2023. Exploring the Limitations of using Large Language Models to Fix Planning Tasks. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Grover, S.; and Mohan, S. 2024. A Demonstration of Natural Language Understanding in Embodied Planning Agents. In *ICAPS 2024 System's Demonstration track*.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Gundawar, A.; Verma, M.; Guan, L.; Valmeekam, K.; Bhambri, S.; and Kambhampati, S. 2024. Robust Planning with LLM-Modulo Framework: Case Study in Travel Planning. *CoRR*, abs/2405.20625.

Guo, W.; Kingston, Z. K.; and Kavraki, L. E. 2024. CaStL: Constraints as Specifications through LLM Translation for Long-Horizon Task and Motion Planning. *CoRR*, abs/2410.22225.

Gupta, A.; Efros, A. A.; and Hebert, M. 2010. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, 482–496. Springer.

Han, M.; Zhu, Y.; Zhu, S.; Wu, Y. N.; and Zhu, Y. 2024. InterPreT: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning. *CoRR*, abs/2405.19758.

Hao, Y.; Zhang, Y.; and Fan, C. 2024. Planning Anything with Rigor: General-Purpose Zero-Shot Planning with LLM-based Formalized Programming. *CoRR*, abs/2410.12112.

Hayton, T.; Porteous, J.; Ferreira, J. F.; and Lindsay, A. 2020. Narrative Planning Model Acquisition from Text Summaries and Descriptions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 1709–1716. AAAI Press.

Hendler, J. A.; Tate, A.; and Drummond, M. 1990. AI Planning: Systems and Techniques. *AI Mag.*, 11(2): 61–77.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15-17 November 2004, Boca Raton, FL, USA*, 294–301. IEEE Computer Society.

Hu, M.; Chen, T.; Zou, Y.; Lei, Y.; Chen, Q.; Li, M.; Mu, Y.; Zhang, H.; Shao, W.; and Luo, P. 2025. Text2World: Benchmarking Large Language Models for Symbolic World Model Generation.

Hu, M.; Zhao, P.; Xu, C.; Sun, Q.; Lou, J.; Lin, Q.; Luo, P.; Rajmohan, S.; and Zhang, D. 2024. AgentGen: Enhancing Planning Abilities for Large Language Model based Agent via Environment and Task Generation. *CoRR*, abs/2408.00764.

Huang, C.; and Zhang, L. 2024. On the Limit of Language Models as Planning Formalizers. *CoRR*, abs/2412.09879.

Huang, J.; Chen, X.; Mishra, S.; Zheng, H. S.; Yu, A. W.; Song, X.; and Zhou, D. 2024a. Large Language Models Cannot Self-Correct Reasoning Yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Huang, R.; Chen, Y.; and Zhang, W. 2014. SAS+ Planning as Satisfiability. *CoRR*, abs/1401.4598.

Huang, S.; Lipovetzky, N.; and Cohn, T. 2024. Planning in the Dark: LLM-Symbolic Planning Pipeline without Experts. *arXiv preprint arXiv:2409.15915*.

Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; and Chen, E. 2024b. Understanding the planning of LLM agents: A survey. *CoRR*, abs/2402.02716.

Ishay, A.; and Lee, J. 2025. LLM+ AL: Bridging Large Language Models and Action Languages for Complex Reasoning about Actions. *arXiv preprint arXiv:2501.00830*.

Izquierdo-Badiola, S.; Canal, G.; Rizzo, C.; and Alenyà, G. 2024. PlanCollabNL: Leveraging Large Language Models for Adaptive Plan Generation in Human-Robot Collaboration. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, 17344–17350. IEEE.

Joublin, F.; Ceravola, A.; Smirnov, P.; Ocker, F.; Deigmoeller, J.; Belardinelli, A.; Wang, C.; Hasler, S.; Tanneberg, D.; and Gienger, M. 2023. CoPAL: Corrective Planning of Robot Actions with Large Language Models. *CoRR*, abs/2310.07263.

Kalland, E. H. 2024. *Enabling Semantic Reasoning in Robots through Natural Language Processing*. Master's thesis, NTNU.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Stechly, K.; Verma, M.; Bhambri, S.; Saldyt, L.; and Murthy, A. 2024. LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. *CoRR*, abs/2402.01817.

Katz, M.; Kokel, H.; Srinivas, K.; and Sohrabi Araghi, S. 2024. Thought of Search: Planning with Language Models Through The Lens of Efficiency. *Advances in Neural Information Processing Systems*, 37: 138491–138568.

Kelly, J.; Calderwood, A.; Wardrip-Fruin, N.; and Mateas, M. 2023. There and Back Again: Extracting Formal Domains for Controllable Neurosymbolic Story Authoring. In Eger, M.; and Cardona-Rivera, R. E., eds., *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, October 08-12, 2023, Salt Lake City, UT, USA*, 64–74. AAAI Press.

Khandelwal, V.; Sheth, A. P.; and Agostinelli, F. 2024. PDDLFuse: A Tool for Generating Diverse Planning Domains. *CoRR*, abs/2411.19886.

Kokel, H.; Katz, M.; Srinivas, K.; and Sohrabi, S. 2024. ACPBench: Reasoning about Action, Change, and Planning. *CoRR*, abs/2410.05669.

Kwon, M.; Kim, Y.; and Kim, Y. J. 2024. Fast and Accurate Task Planning using Neuro-Symbolic Language Models and Multi-level Goal Decomposition. *CoRR*, abs/2409.19250.

Lee, J.; Cantu, M.; Korb, J.; Meth, E.; Griffith, J. D.; Korman, J.; Yuen, A.; Schwartz, P.; and Gertner, A. S. 2024. Planning AI Assistant for Emergency Decision-Making (PlanAID): Framing Planning Problems and Assessing Plans with Large Language Models. In *AAAI 2025 Workshop LM4Plan*.

Li, H.; Chen, Z.; Zhang, J.; and Liu, F. 2024a. LASP: Surveying the State-of-the-Art in Large Language Model-Assisted AI Planning. *CoRR*, abs/2409.01806.

Li, R.; Allal, L. B.; Zi, Y.; Muennighoff, N.; Kocetkov, D.; Mou, C.; Marone, M.; Akiki, C.; Li, J.; Chim, J.; Liu, Q.; Zheltonozhskii, E.; Zhuo, T. Y.; Wang, T.; Dehaene, O.; Davaadorj, M.; Lamy-Poirier, J.; Monteiro, J.; Shliazhko, O.; Gontier, N.; Meade, N.; Zebaze, A.; Yee, M.; Umapathi, L. K.; Zhu, J.; Lipkin, B.; Oblokulov, M.; Wang, Z.; V, R. M.; Stillerman, J.; Patel, S. S.; Abulkhanov, D.; Zocca, M.; Dey, M.; Zhang, Z.; Moustafa-Fahmy, N.; Bhattacharyya, U.; Yu, W.; Singh, S.; Luccioni, S.; Villegas, P.; Kunakov, M.; Zhdanov, F.; Romero, M.; Lee, T.; Timor, N.; Ding, J.; Schlesinger, C.; Schoelkopf, H.; Ebert, J.; Dao, T.; Mishra, M.; Gu, A.; Robinson, J.; Anderson, C. J.; Dolan-Gavitt, B.; Contractor, D.; Reddy, S.; Fried, D.; Bahdanau, D.; Jernite, Y.; Ferrandis, C. M.; Hughes, S.; Wolf, T.; Guha, A.; von Werra, L.; and de Vries, H. 2023. StarCoder: may the source be with you! *CoRR*, abs/2305.06161.

Li, S.; Ma, Z.; Liu, F.; Lu, J.; Xiao, Q.; Sun, K.; Cui, L.; Yang, X.; Liu, P.; and Wang, X. 2024b. Safe Planner: Empowering Safety Awareness in Large Pre-Trained Models for Robot Task Planning. *CoRR*, abs/2411.06920.

Lin, K.; Agia, C.; Migimatsu, T.; Pavone, M.; and Bohg, J. 2023. Text2Motion: from natural language instructions to feasible plans. *Auton. Robots*, 47(8): 1345–1365.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *CoRR*, abs/2304.11477.

Liu, J. X.; Yang, Z.; Idrees, I.; Liang, S.; Schornstein, B.; Tellex, S.; and Shah, A. 2023b. Lang2LTL: Translating Natural Language Commands to Temporal Robot Task Specification. *CoRR*, abs/2302.11649.

Liu, W.; Nie, N.; Mao, J.; Zhang, R.; and Wu, J. 2024a. Learning Compositional Behaviors from Demonstration and Language. In *8th Annual Conference on Robot Learning*.

Liu, Y.; Palmieri, L.; Koch, S.; Georgievski, I.; and Aiello, M. 2024b. DELTA: Decomposed Efficient Long-Term Robot Task Planning using Large Language Models. *CoRR*, abs/2404.03275.

Liu, Y.; Palmieri, L.; Koch, S.; Georgievski, I.; and Aiello, M. 2024c. Towards Human Awareness in Robot Task Planning with Large Language Models. *CoRR*, abs/2404.11267.

Luo, X.; Xu, S.; and Liu, C. 2023. Obtaining hierarchy from human instructions: an llms-based approach. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*.

Lyu, Q.; Havaldar, S.; Stein, A.; Zhang, L.; Rao, D.; Wong, E.; Apidianaki, M.; and Callison-Burch, C. 2023. Faithful Chain-of-Thought Reasoning. *CoRR*, abs/2301.13379.

Mahdavi, S.; Aoki, R.; Tang, K.; and Cao, Y. 2024. Leveraging Environment Interaction for Automated PDDL Generation and Planning with Large Language Models. *CoRR*, abs/2407.12979.

Mao, Y.; Zhang, T.; Cao, X.; Chen, Z.; Liang, X.; Xu, B.; and Fang, H. 2024. NL2STL: Transformation from Logic Natural Language to Signal Temporal Logics using Llama2. In *IEEE International Conference on Cybernetics and Intelligent Systems, CIS 2024, and IEEE International Conference on Robotics, Automation and Mechatronics, RAM 2024, Hangzhou, China, August 8-11, 2024*, 469–474. IEEE.

McDermott, D.; M. Ghallab, A. H.; Knoblock, C.; A. Ram, M. V.; Weld, D.; and Wilkins, D. 1998. Pddl-the planning domain definition language.

Merino, I. P.; and Sabater-Mir, J. 2024. LLM Reasoner and Automated Planner: A New NPC Approach. In Alsinet, T.;

Vilasís-Cardona, X.; García, D.; and Álvarez, E., eds., *Artificial Intelligence Research and Development - Proceedings of the 26th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2024, Barcelona, Spain, 2-4 October 2024*, volume 390 of *Frontiers in Artificial Intelligence and Applications*, 244–247. IOS Press.

Miglani, S.; and Yorke-Smith, N. 2020. Nltopddl: One-shot learning of pddl models from natural language process manuals. In *ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*. ICAPS.

Momennejad, I.; Hasanbeig, H.; Frujeri, F. V.; Sharma, H.; Jojic, N.; Palangi, H.; Ness, R. O.; and Larson, J. 2023. Evaluating Cognitive Maps and Planning in Large Language Models with CogEval. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nabizada, H.; Jeleniewski, T.; Gehlhoff, F.; and Fay, A. 2024. Model-based Workflow for the Automated Generation of PDDL Descriptions. *CoRR*, abs/2408.08145.

Nijkamp, E.; Pang, B.; Hayashi, H.; Tu, L.; Wang, H.; Zhou, Y.; Savarese, S.; and Xiong, C. 2023. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Oates, T.; Alford, R.; Johnson, S.; and Hall, C. 2024. Using Large Language Models to Extract Planning Knowledge from Common Vulnerabilities and Exposures.

Oswald, J. T.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024. Large Language Models as Planning Domain Generators. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, 423–431. AAAI Press.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022. Plansformer: Generating Symbolic Plans using Transformers. *CoRR*, abs/2212.08681.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Srivastava, B.; Horesh, L.; Fabiano, F.; and Loreggia, A. 2023. Understanding the Capabilities of Large Language Models for Automated Planning. *CoRR*, abs/2305.16151.

Pallagani, V.; Muppasani, B. C.; Roy, K.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. P. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, 432–444. AAAI Press.

Pan, J.; Chou, G.; and Berenson, D. 2023. Data-Efficient Learning of Natural Language to Linear Temporal Logic Translators for Robot Task Specification. *CoRR*, abs/2303.08006.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. Y. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, 3806–3824. Association for Computational Linguistics.

Patil, K. 2024. *LLMs for AI planning: a study on error detection and correction in PDDL domain models*. Master's thesis.

Paulius, D.; Agostini, A.; Quartey, B.; and Konidaris, G. 2024. Bootstrapping Object-level Planning with Large Language Models. *arXiv preprint arXiv:2409.12262*.

Puerta-Merino, I.; Núñez-Molina, C.; Mesejo, P.; and Fernández-Olivares, J. 2025. A Roadmap to Guide the Integration of LLMs in Hierarchical Planning. *arXiv preprint arXiv:2501.08068*.

Raman, S. S.; Cohen, V.; Idrees, I.; Rosen, E.; Mooney, R.; Tellex, S.; and Paulius, D. 2024. CAPE: Corrective Actions from Precondition Errors using Large Language Models. arXiv:2211.09935.

Raman, S. S.; Cohen, V.; Rosen, E.; Idrees, I.; Paulius, D.; and Tellex, S. 2022. Planning with Large Language Models via Corrective Re-prompting. *CoRR*, abs/2211.09935.

Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Balog, M.; Kumar, M. P.; Dupont, E.; Ruiz, F. J. R.; Ellenberg, J. S.; Wang, P.; Fawzi, O.; Kohli, P.; and Fawzi, A. 2024. Mathematical discoveries from program search with large language models. *Nat.*, 625(7995): 468–475.

Russell, S.; and Norvig, P. 2020. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson. ISBN 9780134610993.

Sakib, M. S.; and Sun, Y. 2024. Consolidating Trees of Robotic Plans Generated Using Large Language Models to Improve Reliability. *CoRR*, abs/2401.07868.

Sel, B.; Jia, R.; and Jin, M. 2025. LLMs Can Plan Only If We Tell Them. *arXiv preprint arXiv:2501.13545*.

Shah, N. 2024. Autonomously Learning World-Model Representations For Efficient Robot Planning. Technical report, Arizona State University.

Shirai, K.; Beltran-Hernandez, C. C.; Hamaya, M.; Hashimoto, A.; Tanaka, S.; Kawaharazuka, K.; Tanaka, K.; Ushiku, Y.; and Mori, S. 2023. Vision-Language Interpreter for Robot Task Planning. *CoRR*, abs/2311.00967.

Shridhar, M.; Yuan, X.; Côté, M.; Bisk, Y.; Trischler, A.; and Hausknecht, M. J. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sikes, K.; Fine-Morris, M.; Sreedharan, S.; and Roberts, M. 2024a. Traversing the Linguistic Divide: Aligning Semantically Equivalent Fluents Through Model Refinement.

Sikes, K.; Fine-Morris, M.; Sreedharan, S.; Smith, L. N.; and Roberts, M. 2024b. Creating PDDL Models from Javascript using LLMs: Preliminary Results. In *AAAI 2025 Workshop LM4Plan*.

Silver, T.; Hariprasad, V.; Shuttleworth, R. S.; Kumar, N.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. PDDL planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.

Simon, N.; and Muise, C. 2021. A natural language model for generating pddl. In *ICAPS KEPS workshop*.

Singh, I.; Traum, D.; and Thomason, J. 2024. TwoStep: Multi-agent Task Planning using Classical Planners and Large Language Models. *CoRR*, abs/2403.17246.

Singh, S.; Swaminathan, K.; Arora, R.; Singh, R.; Datta, A.; Das, D.; Banerjee, S.; Sridharan, M.; and Krishna, K. M. 2024. Anticipate & Collab: Data-driven Task Anticipation and Knowledge-driven Planning for Human-robot Collaboration. *CoRR*, abs/2404.03587.

Sinha, V. 2024. *Leveraging LLMs for HTN domain model generation via prompt engineering*. Master's thesis.

Smirnov, P.; Joublin, F.; Ceravola, A.; and Gienger, M. 2024. Generating consistent PDDL domains with Large Language Models. *CoRR*, abs/2404.07751.

Stechly, K.; Marquez, M.; and Kambhampati, S. 2023. GPT-4 Doesn't Know It's Wrong: An Analysis of Iterative Prompting for Reasoning Problems. *CoRR*, abs/2310.12397.

Stechly, K.; Valmeekam, K.; and Kambhampati, S. 2024. Chain of Thoughtlessness: An Analysis of CoT in Planning. *CoRR*, abs/2405.04776.

Stein, K.; and Koller, A. 2023. AutoPlanBench: : Automatically generating benchmarks for LLM planners from PDDL. *CoRR*, abs/2311.09830.

Trinh, T. H.; Wu, Y.; Le, Q. V.; He, H.; and Luong, T. 2024. Solving olympiad geometry without human demonstrations. *Nat.*, 625(7995): 476–482.

Tuisov, A.; Vernik, Y.; and Shleyfman, A. 2025. LLM-Generated Heuristics for AI Planning: Do We Even Need Domain-Independence Anymore? *arXiv preprint arXiv:2501.18784*.

Vallati, M.; and Kitchin, D. E., eds. 2020. *Knowledge Engineering Tools and Techniques for AI Planning*. Springer. ISBN 978-3-030-38560-6.

Valmeekam, K.; Marquez, M.; Hernandez, A. O.; Sreedharan, S.; and Kambhampati, S. 2023. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Valmeekam, K.; Marquez, M.; and Kambhampati, S. 2023. Can Large Language Models Really Improve by Self-critiquing Their Own Plans? *CoRR*, abs/2310.08118.

Valmeekam, K.; Stechly, K.; Gundawar, A.; and Kambhampati, S. 2024. Planning in Strawberry Fields: Evaluating and Improving the Planning and Scheduling Capabilities of LRM o1. *arXiv preprint arXiv:2410.02162*.

Valmeekam, K.; Stechly, K.; and Kambhampati, S. 2024. LLMs Still Can't Plan; Can LRMs? A Preliminary Evaluation of OpenAI's o1 on PlanBench. *arXiv preprint arXiv:2409.13373*.

Vyas, K.; Graux, D.; Montella, S.; Vougiouklis, P.; Lai, R.; Li, K.; Ren, Y.; and Pan, J. Z. 2025. An Extensive Evaluation of PDDL Capabilities in off-the-shelf LLMs.

Vygotsky, L. S. 1978. Mind in society: The development of higher psychological processes. *Harvard UP*.

Wang, B.; Wang, Z.; Wang, X.; Cao, Y.; Saurous, R. A.; and Kim, Y. 2023. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wang, E.; Cassano, F.; Wu, C.; Bai, Y.; Song, W.; Nath, V.; Han, Z.; Hendryx, S.; Yue, S.; and Zhang, H. 2024. Planning In Natural Language Improves LLM Search For Code Generation. *CoRR*, abs/2409.03733.

Wang, F.; Lyu, S.; Zhou, P.; Duan, A.; Guo, G.; and Navarro-Alarcon, D. 2025. Instruction-Augmented Long-Horizon Planning: Embedding Grounding Mechanisms in Embodied Mobile Manipulation.

Wei, H.; Zhang, Z.; He, S.; Xia, T.; Pan, S.; and Liu, F. 2025. PlanGenLLMs: A Modern Survey of LLM Planning Capabilities. *arXiv preprint arXiv:2502.11221*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022a. Emergent Abilities of Large Language Models. *CoRR*, abs/2206.07682.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Wong, L.; Mao, J.; Sharma, P.; Siegel, Z. S.; Feng, J.; Korneev, N.; Tenenbaum, J. B.; and Andreas, J. 2023. Learning adaptive planning representations with natural language guidance. *CoRR*, abs/2312.08566.

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024a. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Xie, K.; Yang, I.; Gunerli, J.; and Riedl, M. 2024b. Making Large Language Models into World Models with Precondition and Effect Knowledge. *arXiv preprint arXiv:2409.12278*.

Xie, Y.; Yu, C.; Zhu, T.; Bai, J.; Gong, Z.; and Soh, H. 2023. Translating Natural Language to Planning Goals with Large-Language Models. *CoRR*, abs/2302.05128.

Yang, Z.; Ishay, A.; and Lee, J. 2023. Coupling Large Language Models with Logic Programming for Robust and General Reasoning from Text. In Rogers, A.; Boyd-Graber, J. L.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, 5186–5219. Association for Computational Linguistics.

Ye, R.; Hu, Y.; Bian, Y. A.; Kulm, L.; and Bhattacharjee, T. 2024. MORPHeus: a Multimodal One-armed Robot-assisted Peeling System with Human Users In-the-loop. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, 9540–9547. IEEE.

Ying, L.; Collins, K. M.; Wei, M.; Zhang, C. E.; Zhi-Xuan, T.; Weller, A.; Tenenbaum, J. B.; and Wong, L. 2023. The Neuro-Symbolic Inverse Planning Engine (NIPE): Modeling Probabilistic Social Inferences from Linguistic Inputs. *CoRR*, abs/2306.14325.

Zhang, L. 2024. Structured Event Reasoning with Large Language Models. *CoRR*, abs/2408.16098.

Zhang, L.; Jansen, P.; Zhang, T.; Clark, P.; Callison-Burch, C.; and Tandon, N. 2024a. PDDLEGO: Iterative Planning in Textual Environments. *CoRR*, abs/2405.19793.

Zhang, T.; Zhang, L.; Hou, Z.; Wang, Z.; Gu, Y.; Clark, P.; Callison-Burch, C.; and Tandon, N. 2024b. PROC2PDDL: Open-Domain Planning Representations from Texts. *CoRR*, abs/2403.00092.

Zhang, X.; Altaweel, Z.; Hayamizu, Y.; Ding, Y.; Amiri, S.; Yang, H.; Kaminski, A.; Esselink, C.; and Zhang, S. 2024c. DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning. *CoRR*, abs/2406.17659.

Zhang, X.; Qin, H.; Wang, F.; Dong, Y.; and Li, J. 2024d. LaMMA-P: Generalizable Multi-Agent Long-Horizon Task Allocation and Planning with LM-Driven PDDL Planner. *CoRR*, abs/2409.20560.

Zhao, Z.; Cheng, S.; Ding, Y.; Zhou, Z.; Zhang, S.; Xu, D.; and Zhao, Y. 2024. A Survey of Optimization-based Task and Motion Planning: From Classical To Learning Approaches. *CoRR*, abs/2404.02817.

Zheng, H. S.; Mishra, S.; Zhang, H.; Chen, X.; Chen, M.; Nova, A.; Hou, L.; Cheng, H.; Le, Q. V.; Chi, E. H.; and Zhou, D. 2024. NATURAL PLAN: Benchmarking LLMs on Natural Language Planning. *CoRR*, abs/2406.04520.

Zhou, Z.; Song, J.; Yao, K.; Shu, Z.; and Ma, L. 2023. ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning. *CoRR*, abs/2308.13724.

Zuo, M.; Velez, F. P.; Li, X.; Littman, M. L.; and Bach, S. H. 2024. Planetarium: A Rigorous Benchmark for Translating Text to Structured Planning Languages. *CoRR*, abs/2407.03321.

A Additional Information on PDDL

A.1 Why PDDL?

PDDL, as a **declarative** language, differs fundamentally from **imperative** languages like Python in how problems are expressed and solved. Declarative programming specifies *what* needs to be achieved rather than *how* to achieve it, leaving execution to an external planner. In contrast, imperative programming defines explicit step-by-step instructions, requiring precise control over execution flow. While LLMs struggle with logical reasoning in imperative programming due to its sequential dependencies, they can perform well with declarative representations like PDDL. Using LLMs for PDDL model construction is **not traditional code generation** but rather knowledge structuring—organizing states, actions, and constraints into a formalized model.

A.2 PDDL Example

Automated Planning is a specialized field within AI that can be challenging for those unfamiliar with its principles. A key tool in classical planning is the Planning Domain Definition Language (PDDL), which models planning problems and domains. We illustrate its concepts with the Blocksworld problem, where blocks must be stacked in a specific order using actions like picking up, unstacking, and placing blocks, all while respecting constraints like moving only one block at a time or not disturbing stacked blocks.

Demonstrated below is the Blocksworld PDDL domain file $\mathbb{DF}$:

```
(define (domain blocksworld)
    (:requirements :strips)
    (:predicates (clear ?x) (on-table ?x) (arm-empty) (holding ?x) (on ?x ?y))
    (:action pickup
        :parameters (?ob)
        :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
        :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob)) (not
            (arm-empty)))
    ))
```

Predicates define relationships or properties that can be true or false, such as `(on ?x ?y)` for block `?x` on `?y`, `(ontable ?x)` for `?x` on the table, `(clear ?x)` for `?x` having nothing on top, and `(holding ?x)` for the robot holding `?x`. **Actions** describe possible state changes. For instance, `(pick-up)` contains the *parameter(s)* block `?ob`, *preconditions* requiring `(clear ?ob)` and `(ontable ?ob)`, and *effects* updating the state to reflect the robot holding `?ob`, which is no longer on the table and clear.

The following is the corresponding PDDL problem/task file $\mathbb{PF}$:

```
(define (problem blocksworld-problem)
   (:domain blocksworld)
   (:objects A B C) ; Blocks
   (:init (ontable A) (ontable B) (on C A) (clear B) (clear C)) ; Initial state
   (:goal (and (on A B) (on B C)))) ; Goal state
```

Objects represent the entities involved, such as blocks A, B, and C. The **initial state** defines the starting arrangement, where blocks A and B are on the table, block C is on A, and both B and C are clear. The **goal state** specifies the desired configuration, where block A is stacked on B, and block B is stacked on C.

Given the above PDDL domain and problem, a classical planner might generate the following plan:

```
Unstack C from A
Put C on table
Pick up A
Stack A on B
Pick up B
Stack B on C
```

B (L2P) Framework

B.1 General Library Overview

L2P provides a comprehensive suite of tools for PDDL model creation and validation. The Builder classes enable users to prompt the LLM to generate essential components of a PDDL domain, such as types, predicates, and actions, along with their parameters, preconditions, and effects. It also supports task specification, including objects, initial, and goal states that correspond to the given domain. L2P features a customized Feedback Builder class that incorporates both LLM-generated feedback and human input, or a combination of the two. Additionally, the library includes a syntax validation tool that detects common PDDL syntax errors, using this feedback to improve the accuracy of the generated models—as illustrated in Figure 6, which shows how LLM feedback can be used to refine a PDDL problem specification. L2P is now publicly available for pip installation and can be found at: **https://github.com/AI-Planning/l2p**

B.2 L2P Usage Example

Below are example usages using our L2P library. Full documentation can be found on our website.

```
import os
from l2p import *

domain_builder = DomainBuilder() # initialize Domain Builder class

# REPLACE WITH OWN API KEY
api_key = os.environ.get('OPENAI_API_KEY')
llm = OPENAI(model="gpt-4o-mini", api_key=api_key)

# retrieve prompt information
base_path='tests/usage/prompts/domain/'
domain_desc = load_file(f'{base_path}blocksworld_domain.txt')
extract_predicates_prompt = load_file(f'{base_path}extract_predicates.txt')
types = load_file(f'{base_path}types.json')
action = load_file(f'{base_path}action.json')

# extract predicates via LLM
predicates, llm_output = domain_builder.extract_predicates(
    model=llm,
    domain_desc=domain_desc,
    prompt_template=extract_predicates_prompt,
    types=types,
    nl_actions={action['action_name']: action['action_desc']}
    )

# format key info into PDDL strings
predicate_str = "\n".join([pred["clean"].replace(":", " ; ") for pred in predicates])

print(f"PDDL domain predicates:\n{predicate_str}")

-------------------------------------------------------------------------------------


### OUTPUT
(holding ?a - arm ?b - block) ;  true if the arm ?a is holding the block ?b
(on_top ?b1 - block ?b2 - block) ;  true if the block ?b1 is on top of the block ?b2
(clear ?b - block) ;  true if the block ?b is clear (no block on top of it)
(on_table ?b - block) ;  true if the block ?b is on the table
(empty ?a - arm) ;  true if the arm ?a is empty (not holding any block)
```

Figure 4: L2P usage - generating simple PDDL predicates

```
import os
from l2p import *

task_builder = TaskBuilder() # initialize Task Builder class
api_key = os.environ.get('OPENAI_API_KEY')
llm = OPENAI(model="gpt-4o-mini", api_key=api_key)

# load in assumptions
problem_desc = load_file(r'tests/usage/prompts/problem/blocksworld_problem.txt')
extract_task_prompt = load_file(r'tests/usage/prompts/problem/extract_task.txt')
types = load_file(r'tests/usage/prompts/domain/types.json')
predicates_json = load_file(r'tests/usage/prompts/domain/predicates.json')
predicates: List[Predicate] = [Predicate(**item) for item in predicates_json]

# extract PDDL task specifications via LLM
objects, initial_states, goal_states, llm_response = task_builder.extract_task(
    model=llm,
    problem_desc=problem_desc,
    prompt_template=extract_task_prompt,
    types=types,
    predicates=predicates
    )

# format key info into PDDL strings
objects_str = task_builder.format_objects(objects)
initial_str = task_builder.format_initial(initial_states)
goal_str = task_builder.format_goal(goal_states)

# generate task file
pddl_problem = task_builder.generate_task(
    domain="blocksworld",
    problem="blocksworld_problem",
    objects=objects_str,
    initial=initial_str,
    goal=goal_str)

print(f"### LLM OUTPUT:\n {pddl_problem}")
-------------------------------------------------------------------------------------
### LLM OUTPUT
(define
   (problem blocksworld_problem)
   (:domain blocksworld)
   (:objects
      blue_block - object
      red_block - object
      yellow_block - object
      green_block - object
   )
   (:init
      (on_top blue_block red_block)
      (on_top red_block yellow_block)
      (on_table yellow_block)
      (on_table green_block)
      (clear blue_block) (clear yellow_block) (clear green_block)
   )
   (:goal
      (and (on_top red_block green_block))
   )
)
```

Figure 5: L2P usage - generating simple PDDL task specification

```
import os
from l2p import *

feedback_builder = FeedbackBuilder()
api_key = os.environ.get('OPENAI_API_KEY')
llm = OPENAI(model="gpt-4o-mini", api_key=api_key)

problem_desc = load_file(r'tests/usage/prompts/problem/blocksworld_problem.txt')
types = load_file(r'tests/usage/prompts/domain/types.json')
feedback_template = load_file(r'tests/usage/prompts/problem/feedback.txt')
predicates_json = load_file(r'tests/usage/prompts/domain/predicates.json')
predicates: List[Predicate] = [Predicate(**item) for item in predicates_json]
llm_response = load_file(r'tests/usage/prompts/domain/llm_output_task.txt')

objects, initial, goal, feedback_response = feedback_builder.task_feedback(
    model=llm,
    problem_desc=problem_desc,
    feedback_template=feedback_template,
    feedback_type="llm",
    predicates=predicates,
    types=types,
    llm_response=llm_response)

print("FEEDBACK:\n", feedback_response)
-------------------------------------------------------------------------------------
### LLM OUTPUT
FEEDBACK:
1. Are any necessary objects missing?
   All necessary objects are included based on the problem description. So: No.
2. Are any unnecessary objects included?
   All objects included are relevant to the problem. Hence: No.
3. Are any objects defined with the wrong type?
   All objects are correctly defined as "object". Therefore: No.
4. Are any unnecessary or incorrect predicates declared?
   All predicates used in the initial state are relevant and correctly applied. Thus: No.
5. Are any needed or expected predicates missing from the initial state?
   The initial state is missing the predicate for the red block being clear. Since the red block is covered by
   the blue block, it should not be clear. Therefore: Yes.
6. Is anything missing from the goal state?
   The goal state accurately reflects the desired outcome of having the red block on top of the green block.
        So: No.
7. Is anything unnecessary included in the goal description?
   The goal description is concise and only includes what is necessary. Therefore: No.
8. Should any predicate be used in a symmetrical manner?
   The predicates used do not require symmetry as they are directional in nature. So: No.

My concrete suggestions are the following:
- Add the predicate to indicate that the red block is not clear:
  - (clear red_block) should be removed from the initial state since the red block is covered by the blue
      block.

Final output should reflect this change in the initial state:
```
(clear blue_block): blue block is clear
(clear yellow_block): yellow block is clear
(clear green_block): green block is clear
```

Overall, the feedback is: Yes, the initial state needs to reflect that the red block is not clear.
```

Figure 6: L2P usage - generating LLM-feedback on task specification

```
## PREDICATES
{'name': 'truck-at',
    'desc': 'true if the truck ?t is currently at location ?l',
    'raw': '(truck-at ?t - truck ?l - location): true if the truck ?t is currently at location ?l',
    'params': OrderedDict([('?t', 'truck'), ('?l', 'location')]),
    'clean': '(truck-at ?t - truck ?l - location): true if the truck ?t is currently at location ?l'}
{'name': 'package-at',
    'desc': 'true if the package ?p is currently at location ?l',
    'raw': '(package-at ?p - package ?l - location): true if the package ?p is currently at location ?l',
    'params': OrderedDict([('?p', 'package'), ('?l', 'location')]),
    'clean': '(package-at ?p - package ?l - location): true if the package ?p is currently at location ?l'}
{'name': 'truck-holding',
    'desc': 'true if the truck ?t is currently holding the package ?p',
    'raw': '(truck-holding ?t - truck ?p - package): true if the truck ?t is currently holding the package ?p',
    'params': OrderedDict([('?t', 'truck'), ('?p', 'package')]),
    'clean': '(truck-holding ?t - truck ?p - package): true if the truck ?t is currently holding the package ?p'}
{'name': 'truck-has-space',
    'desc': 'true if the truck ?t has space to load more packages',
    'raw': '(truck-has-space ?t - truck): true if the truck ?t has space to load more packages',
    'params': OrderedDict([('?t', 'truck')]),
    'clean': '(truck-has-space ?t - truck): true if the truck ?t has space to load more packages'}
{'name': 'plane-at',
    'desc': 'true if the airplane ?a is located at location ?l',
    'raw': '(plane-at ?a - plane ?l - location): true if the airplane ?a is located at location ?l',
    'params': OrderedDict([('?a', 'plane'), ('?l', 'location')]),
    'clean': '(plane-at ?a - plane ?l - location): true if the airplane ?a is located at location ?l'}
{'name': 'plane-holding',
    'desc': 'true if the airplane ?a is currently holding the package ?p',
    'raw': '(plane-holding ?a - plane ?p - package): true if the airplane ?a is currently holding package ?p',
    'params': OrderedDict([('?a', 'plane'), ('?p', 'package')]),
    'clean': '(plane-holding ?a - plane ?p - package): true if the airplane ?a is currently holding package ?p'}
{'name': 'connected-locations',
    'desc': 'true if location ?l1 is directly connected to location ?l2 in city ?c',
    'raw': '(connected-locations ?l1 - location ?l2 - location ?c - city): ?l1 is connected to ?l2 in city ?c',
    'params': OrderedDict([('?l1', 'location'), ('?l2', 'location'), ('?c', 'city')]),
    'clean': '(connected-locations ?l1 - location ?l2 - location ?c - city): ?l1 is connected to ?l2 in city ?c'}
```

Figure 7: L2P formatted predicate output from LLM (action-by-action algorithm) of Logistics domain.

C Paper Overview

This section contains an overview of the core framework papers found within this survey. The table provides a structured distinction between the papers based on their methodological approaches and specific areas of focus, such as what aspect are they modeling, prompting style/weight, and feedback mechanism. Additionally, it highlights papers already covered by the L2P library, ensuring users have quick access to resources that have been implemented or analyzed in detail. These covered papers will be included in our GitHub repository for easy reference. The distinction in the table also aims to emphasize gaps in the literature, such as under explored techniques or novel applications of NL-PDDL through LLMs, paving the way for future research. To maintain relevance, we plan to keep an updated installment of all NL-PDDL works leveraging LLMs, ensuring this survey evolves alongside the field.

Framework	Task	Domain	NHI	Prompt	Feedback
*(Collins et al. 2022)	✓	×	✓	Medium (Few-shot)	None
(Xie et al. 2023)	✓	×	✓	Medium (Few-shot)	None
(Lyu et al. 2023)	✓	×	✓	Light (CoT)	None
(Grover and Mohan 2024)	✓	×	✓	Medium (Few-shot)	Env.
(Lin et al. 2023)	✓	×	✓	Medium (Few-shot)	LLM+Env.
*(Liu et al. 2023a)	✓	×	✓	Heavy (Few-shot)	None
(Birr et al. 2024)	✓	×	✓	Light (CoT)	External tool
(Agarwal and Sreepathy 2024)	✓	×	×	Medium	None
(Dagan, Keller, and Lascarides 2023)	✓	×	✓	Medium	LLM + Env.
(Zhang et al. 2024a)	✓	×	✓	Medium (Few-shot)	LLM
(Liu et al. 2024c)	✓	×	×	Extended Scene Graph	Env.
(Singh et al. 2024)	✓	×	×	Light	LLM+Deviation
(Izquierdo-Badiola et al. 2024)	✓	×	✓	Light	LLM
(Singh, Traum, and Thomason 2024)	✓	×	✓	Medium (One-shot+)	LLM Helper Agent
(Zhang et al. 2024d)	✓	×	✓	Light	FastDownward+External Tool
(Oates et al. 2024)	×	✓	×	Medium (Few-shot, in-context)	Human
*(Zhang et al. 2024b)	×	✓	×	Medium (Few-shot)	None
*(Guan et al. 2023)	×	✓	×	Light	Human
(Huang, Lipovetzky, and Cohn 2024)	×	✓	✓	Light	Sentence Encoder Filter
(Wong et al. 2023)	×	✓	✓	Medium (Few-shot)	None
(Liu et al. 2024a)	×	✓	×	Heavy	Human
(Ding et al. 2023a)	×	✓	✓	Light	LLM
(Chen et al. 2024)	×	✓	✓	Medium	Env.
(Kelly et al. 2023)	✓	✓	✓	Medium (One-shot+)	External tool
(Smirnov et al. 2024)	✓	✓	✓	Light	LLM
(Liu et al. 2024b)	✓	✓	✓	Medium (One-shot+)	None
(Zhou et al. 2023)	✓	✓	✓	Heavy (Few-shot)	None
(Ye et al. 2024)	✓	✓	×	Heavy	Human
(Han et al. 2024)	✓	✓	×	Light	Human
*(Gestrin, Kuhlmann, and Seipp 2024)	✓	✓	✓	Heavy (Few-shot)	LLM + Human
(da Silva et al. 2024)	✓	✓	✓	Light	LLM
(Mahdavi et al. 2024)	✓	✓	✓	Light	LLM + Env.
(Sakib and Sun 2024)	✓	✓	✓	Light	None
(Ying et al. 2023)	✓	✓	✓	Medium (Few-shot)	Syntax rejection filter

Table 1: Summary of frameworks in this survey. *NHI* = No Human Intervention. *Papers reconstructed by L2P