

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

Marcus Tantakoun^{1,2}, Christian Muise^{1,2}, Xiaodan Zhu^{1,3}

¹Ingenuity Labs Research Institute, Queen’s University

²School of Computing, Queen’s University

³Department of Electrical and Computer Engineering, Queen’s University

{20mt1, christian.muise, xiaodan.zhu}@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 and introduces new challenges. This paper aims to provide a timely survey of the current research with an in-depth analysis, positioning LLMs as tools for formalizing and refining planning specifications to support reliable off-the-shelf 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.

1 Introduction

The advent of 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 a significant bottleneck (Bengio, 2020). Furthermore, LLMs face challenges with long-term planning and reasoning, often producing unreliable plans (Valmeekam et al., 2024b; Pallagani et al., 2023; Momennejad et al., 2023), frequently failing to account for the effects and requirements of actions as they scale (Stechly et al., 2024), with performance degrading under self-iterative LLM feedback (Stechly et al., 2023; Valmeekam et al., 2023b; Huang et al., 2024a).

State-of-the-art LLMs have shown limited planning capability by directly generating action

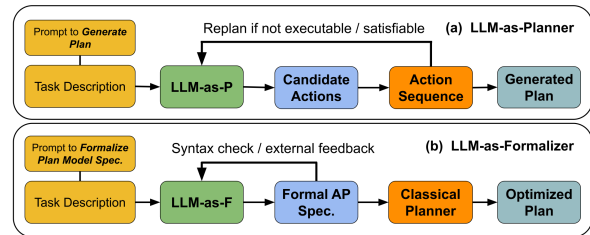


Figure 1: Distinction of planning using LLMs: (a) LLM-as-Planner uses LLMs for direct I/O planning; (b) LLM-as-Formalizer generates planning specifications for existing task planning methods (i.e. PDDL).

sequences—as the correctness, optimality, and reliability of their outputs are not guaranteed. Classical Automated Planning (AP) synthesizes plans through structured representation, logic, and search methods which are not subject to the weaknesses mentioned. Meanwhile, LLMs possess promising capabilities at extracting, interpreting, and refining planning model specifications from natural language (NL), acting as complementary components that can enable classical planners to generate robust solutions. Bringing the advantages of AP and LLMs together—LLMs for constructing planning specifications, AP systems for execution—defines the focus of this survey, a paradigm we call **LLMs-as-Formalizers** for constructing AP models.

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 et al., 2024b). (ii) **Adaptability**: LLMs can aid systems in adapting to dynamic environments to capture real-world

nuances better, reducing the need for manual edits (Dagan et al., 2023). (iii) **Agnostic Modeling**: LLMs trained on large corpora of diverse data can generalize across various domains without needing task-specific tuning, which reduces the reliance on specialized expertise, a major bottleneck in real-world AP integration (Gestrin et al., 2024).

Our survey’s taxonomy is divided into three key areas: **Model Generation**, the process of extracting natural language input—originating from users or the environment—into structured planning model formalizations, 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 Editing** (sec. 3.2) systematically addresses code refinement and the repair of errors and inconsistencies in ill-defined planning formalizations. Finally, **Model Benchmarks** (sec. 3.3) encompass assessments of both the performance of LLMs in planning tasks and the quality of LLM-generated planning formalizations.

To our knowledge, this is the first comprehensive survey of LLM-driven AP-model specification. 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 for a comprehensive field overview.
- A summary of both shared and novel technical approaches 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 $s \in \mathcal{S}$ 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 , which consists of a sequence of actions $\langle a_1, a_2, \dots, a_n \rangle$ 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 .

2.2 Planning Domain Definition Language

As a fundamental building block of AP, the Planning Domain Definition Language (PDDL) (McDermott et al., 1998) is one of the most widely used formalisms for encoding planning tasks. These PDDL models serve as formal AP specifications, defining structured symbolic blueprints that enable off-the-shelf external planners to generate robust and optimized solutions. A PDDL model is composed 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 the 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 states $s^{\mathcal{I}}$ and goal conditions $s^{\mathcal{G}}$. We provide a concrete example in Appendix B. The standardization and use of PDDL in planning have strongly facilitated sharing and benchmarking. This allows a wide selection of tools to support the validation and refinement of code. 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.

2.3 Large Language Models + Planning

The advancements of Large Language Models (LLMs) have shown promise in generating highly structured outputs, such as executable code, from NL descriptions (Li et al., 2023; Wang et al., 2024; Nijkamp et al., 2023). PDDL-LLM research is

recent, with initial studies in (Miglani and Yorke-Smith, 2020; Feng et al., 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. (2024c) survey a limited amount of papers to compose their high-level abstraction of LLM-augmented planning agents. Pallagani et al. (2024) go beyond the scope of traditional AP, encapsulating broader constructs, whereas Zhao et al. (2024) provide an extensive overview of LLM-TAMP applications. The work most akin to our paper is (Li et al., 2024a), which reviews studies using LLMs in planning with PDDL; however, their survey mainly consists of works comprising *LLMs-as-Planners* (cf. Figure 1).

Scope of Survey: LLMs+AP is an expansive field encompassing many research areas, making it impractical to cover its entirety in a single survey with details and insights. Broadly, LLMs+AP paradigms can be categorized as: (i) *LLMs-as-Heuristics*, where LLMs enhance search efficiency via heuristic guidance (Silver et al., 2022; Hirsch et al., 2024; Tuisov et al., 2025; Sel et al., 2025); (ii) *LLMs-as-Planners*, where they either directly analyze action sequences (Zhang et al., 2024c; Lin et al., 2023) or propose plans that are refined through post-hoc methods (Gundawar et al., 2024; Arora and Kambhampati, 2023; Pallagani et al., 2022; Burns et al., 2024). In contrast, our survey analyzes (iii) *LLMs-as-Formalizers*, a paradigm in which LLMs are leveraged to construct AP models. Specifically, LLMs assist in defining planning model specifications, supported by domain-independent planners to generate solutions. Our paper surveys approximately 80 existing works, which utilize LLMs to construct planning models, discussing research questions that drive potential directions.

3 LLMs for Constructing Automated Planning Models

We consider the research on leveraging powerful LLMs to assist in constructing planning models to be of critical importance. Verifiable planning modules remain the backbone of planning, ensuring reliability, robustness, and explainability. Note that

deploying LLMs themselves in an end-to-end manner to perform planning still falls short of providing soundness guarantees (Valmeekam et al., 2024a) and may have principled weaknesses. We organize the existing works into a taxonomy comprising three key areas: **Model Generation**, **Model Editing**, and **Model Benchmarks**—where the term **model**, in this context, refers to AP specifications such as PDDL, as illustrated in Figure 2. The tasks require joint efforts from the NLP and automated planning community.

3.1 Model Generation

A large portion of this survey focuses on **Model Generation**—extracting and formalizing planning specifications from the user or environment via natural language input. This is further divided into three aspects: *Task Modeling* (sec. 3.1.1) defines objectives as initial conditions and goal states; *Domain Modeling* (sec. 3.1.2) defines the foundational components like entities, actions, and relationships in the system; and *Hybrid Modeling* (sec. 3.1.3) integrates both aspects to create a complete model, enabling end-to-end planning. A summary of core frameworks is provided in Appendix A.

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

- RQ1:** How can LLMs accurately align with human goals, ensuring these planning model specifications correctly represent desired expectations and objectives?
- RQ2:** To what extent and granularity of detail can NL instructions be effectively translated into accurate planning 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. Kwon et al. (2024) decompose 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

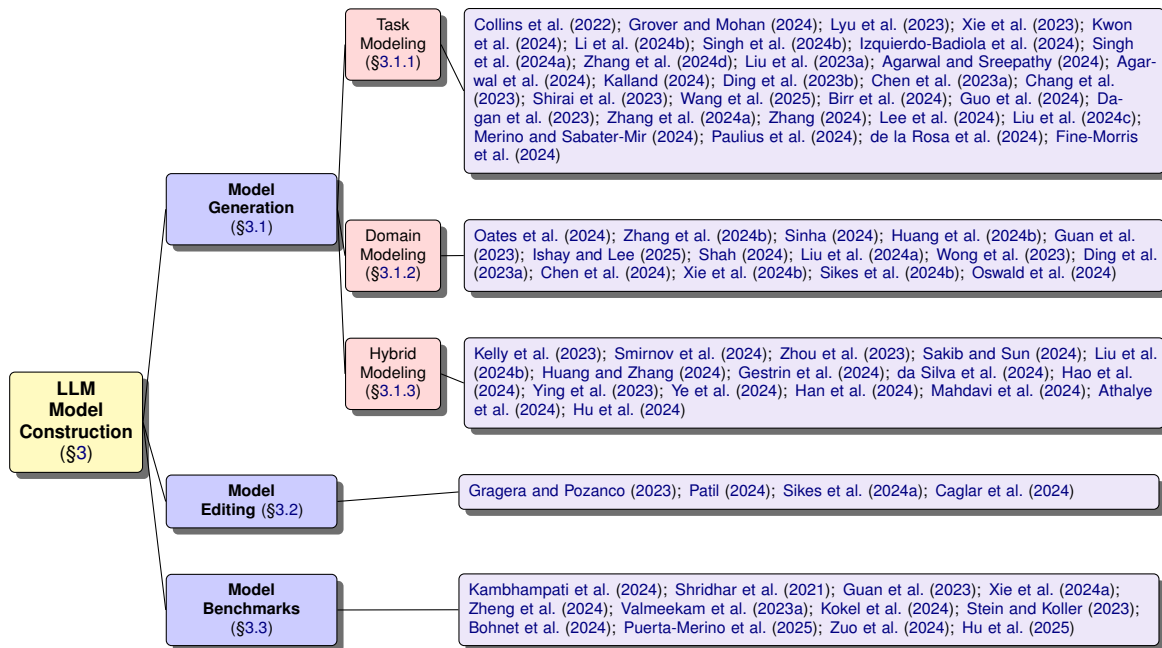


Figure 2: Taxonomy of research in LLM Planning Model Specification

a closed-loop VLM-planner to operate based on real-time environmental observations.

Branching to *multi-agent goal collaboration*, **DaTAPlan** (Singh et al., 2024b) 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 agents which are then encoded into PDDL, translating LLM output sub-goals into PDDL goals, and modifying the action costs based on LLM recommendations. **TwoStep** (Singh et al., 2024a) decompose multi-agent planning problems into two single-agent problems with mechanisms to ensure smooth coordination. **LaMMA-P** (Zhang et al., 2024d) uses LLMs to 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. In open-loop systems, **LLM+P** (Liu et al., 2023a) uses in-context (IC) examples of a related NL problem and its PDDL representation 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. **Kalland** (2024) combines a language and an automatic speech recognition model to generate

PDDL instances. Recent work converts NL instructions into structured geometric representations that bridge abstract language understanding and spatial reasoning for task and motion planning (Ding et al., 2023b; Chen et al., 2023a; Chang et al., 2023), while other approaches integrate LLMs with Vision Language Models (VLMs) for visual perception to further ground language understanding in spatial contexts (Shirai et al., 2023; Wang et al., 2025).

In closed-loop systems, **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 et al. (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 et al., 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. In terms of human-in-the-loop collaboration, (Lee et al., 2024) introduce **PlanAID**, a system that uses Retrieval-Augmented Generation (RAG) to assist LLMs in generating emergency operation plans (EOPs) through improved user interaction. **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.

Rather than extracting PDDL directly, Paulius et al. (2024) 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. Fine-Morris et al. (2024) decompose NL goals into predicate-based Python dictionaries, which are then formatted into HDDL decomposition methods. Beyond PDDL, LLM+AP has expanded generating other planning specifications (Pan et al., 2023) such as *temporal logic* (TL) representations (Chen et al., 2023b), including TSL (Murphy et al., 2024), STL (Mao et al., 2024), and LTL (Cosler et al., 2023; Liu et al., 2023b; Manas et al., 2024; Luo et al., 2023). Recent work shows that LLMs can define the *task search space* by generating successor and goal state Python functions, enabling classical solvers to explore the space efficiently (Katz et al., 2024; Cao et al., 2024).

Summary and Future Directions: Some methods directly translate NL task descriptions into PDDL (Kelly et al., 2023). Others enhance goal specification by incorporating reasoning chains and few-shot examples (Lyu et al., 2023; Liu et al., 2023a). However, these current approaches rely on an explicit mapping between NL and PDDL code, limiting their processes as code translation tasks. 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 single query. 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 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). Sinha (2024) proposes a structured prompt engineering approach to generate domain models in the Hierarchical Planning Definition Language (HPDL). Huang et al. (2024b) use multiple LLM-generated candidate PDDL action schemas, which are then passed through a sentence encoder to compute the semantic relatedness of code and original NL descriptions. Following the candidate filtering approach, pix2pred (Athalye et al., 2024) leverages VLMs to propose predicates and determine their truth values in demonstrations.

Guan et al. (2023) recognize the impracticality of LLMs generating fully functioning PDDL domains in a single call (Kambhampati et al., 2024). Their framework LLM+DM (Domain Model) 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 in BC+ syntax (extension of Answer Set Programming), while (Sikes et al., 2024b) translates Javascript functions to PDDL in incremental stages. Shah (2024) presents LAMP, an extensive series of proposed algorithms that learn abstract PDDL domain models. BLADE (Liu et al., 2024a) bridges language-annotated human demonstrations and primitive action interfaces by tasking an LLM to define the PDDL action model preconditions and effects conditioned on behaviors containing all possible sequences of contact primitive and other behaviors preceding it.

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.

To evaluate domain quality, Oswald et al. (2024) addresses limitations of manual human evaluation (Hayton et al., 2020; Huang et al., 2014) and string-based comparison methods that assess similarity to ground truth. 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 and Future Directions: Kambhampati et al. (2024); Wong et al. (2023) use incremental methods that iteratively refine models. Real-world examples have shown to enhance contextual output 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, by generating and relying on a single domain model, current methods risk rendering the entire planning process invalid if that model fails to capture implicit user constraints. Future approaches should consider generating multiple candidate domains—or specific components, such as predicate definitions—to better accommodate ambiguity and uncertainty in user intent (Huang et al., 2024b; Athalye et al., 2024).

3.1.3 Hybrid Modeling

Hybrid modeling combines PDDL domain and problem systems. 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. **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 et al., 2004). Sakib and Sun (2024) generate multiple high-level task plans in Knowledge Graphs (KG), prunes unnecessary 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 et al., 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. 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. **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 et al., 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.

For real-world grounding, **MORPHEus** (Ye et al., 2024) focuses 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. 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.

Instead of generating models for external planners, **AgentGen** (Hu et al., 2024) uses LLMs to synthesize diverse PDDL tasks and NL descriptions for training LLM-based agents. Their work demonstrate that instruction-tuned models show significant gains in both in-domain and out-of-domain planning tasks. Hu et al. (2025) further fine-tune *LLaMA-3.1* on this dataset, finding notable improvements in domain model generation, especially for larger models.

Summary and Future Directions: Complexities arise when coordinating the domain and respective 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 et al., 2024). These linear pipelines risk cascading errors, such as the possibility of new objects in the later stages of 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

The use of LLMs serving more as assistive tools than fully autonomous generative solutions has shown promising applications for LLM+AP integration. Understanding LLM-editing decisions in refining specifications can support authors 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. 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 LLMs excel at syntax correction but are less reliable with semantic inconsistencies. Sikes et al. (2024a) addresses planning model failures caused by semantically equivalent but syntactically distinct

state variables, a common issue when integrating information from heterogeneous sources. Their approach introduces meta-actions to bridge these mismatches and iteratively refines the model to ensure valid plan generation. Caglar et al. (2024) address the challenge of modifying model spaces beyond classical planning by evaluating how effectively LLMs generate plausible model edits—especially to fix unsolvability and plan executability—to support combinatorial search and manual methods.

Summary and Future Directions: Current research shows promise in using LLMs to correct syntactic errors, but addressing semantic errors remains a significant challenge (Patil, 2024) leading to non-executable or semantically inconsistent plans. Future work should explore post-hoc correction strategies. For instance, researchers could explore strategies to analyze plan outputs, identifying semantic inconsistencies through automated metrics or human evaluation systems as grounded feedback for error-correction.

3.3 Model Benchmarks

LLMs, with their non-deterministic output behaviors, 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).

LLM+AP benchmarks typically fall into two categories: (1) evaluating LLMs’ direct planning abilities; (2) assessing the quality of planning specifications produced by LLMs. While this survey focuses on the latter, we recognize that end-to-end planning benchmarks can also support research on LLM-generated models for external planners.

LLMs-as-Planner Benchmarks: 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 et al., 2010) planning problem by altering the named types so they are semantically equivalent but syntactically nonsensical. **ALFWorld** (Shridhar et al., 2021) and **Household** Guan et al. (2023) tackles the complexities of real-world typical household environment that

uses PDDL semantics to produce textual observations and high-level actions. **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. **PlanBench** (Valmeekam et al., 2023a) 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. **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) introduce a scalable benchmark suite in both PDDL and natural language to evaluate LLMs across diverse planning strategies, along with a method for translating PDDL benchmarks into natural language. Puerta-Merino et al. (2025) propose a road map and benchmark to address the gap of LLM integration in Hierarchical Planning (HP).

LLMs-as-Planning-Formalizers Benchmarks: **Planetarium** (Zuo et al., 2024) provides a rigorous benchmark for evaluating PDDL task/problems produced by LLMs, highlighting two key 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. However, it currently only supports Blocksworld, Gripper, and Floor Tile domains—well-known but narrow in dataset variability. 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. Key metrics, including executability, structural similarity, and component-wise F1 scores, are employed while exploring state-of-the-art LLMs and fine-tuning techniques. However, the reliance on executability as a gating metric excludes non-executable domains from component-wise scoring, causing minor syntax errors to skew overall quality assessments.

Summary and Future Directions: Assessing the quality of LLM-generated PDDL models (Zuo et al., 2024; Hu et al., 2025; 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). Future work should explore solutions for establishing dynamic benchmark standards for domains, actively involving the planning community in its ongoing refinement. Khandelwal et al. (2024) proposes a tool for generating diverse and complex planning domains, which could serve as a foundation for such a benchmark.

4 Language-to-Plan (L2P)

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 ensures soundness via iterative plan refinement with external verifiers, shifting focus from direct planning to PDDL generation with integrated verifiers and user-guided refinement through complete, planner-executable specifications. 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 fully autonomous end-to-end pipelines, reducing the need for manual authoring.

¹Code made publicly at: <https://github.com/AI-Planning/l2p>

```

1 def run_aba_alg(
2     model, action_model, domain_desc, hier,
3     prompt, max_iter: int=2
4 ) -> tuple[list[Predicate], list[Action]]:
5
6     actions = list(action_model.keys())
7     pred_list = []
8     for _ in range(max_iter):
9         action_list = []
10        for _, action in enumerate(actions):
11            # extract action/predicates (L2P)
12            pddl_action, new_preds, _, _ = (
13                builder.formalize_pddl_action(
14                    model=model,
15                    domain_desc=domain_desc,
16                    prompt_template=prompt,
17                    action_name=action,
18                    action_desc=action_model[
19                        action][ 'desc' ],
20                    types=hier[ "hierarchy" ],
21                    predicates=pred_list,
22                    extract_new_preds=True
23                )
24            )
25            pred_list.extend(new_preds)
26            action_list.append(pddl_action)
27
28    pred_list = prune_predicates(pred_list,
29                                action_list)
30
31    return pred_list, action_list

```

Figure 3: A shortened L2P reconstruction of the ‘action-by-action algorithm’ (Guan et al., 2023), which iteratively generates PDDL actions while updating a dynamic predicate list. Output found in Figure 8.

Appendix C demonstrates examples of L2P usage. To demonstrate the flexibility of the framework, L2P re-implements some key papers covered in this survey (refer to Appendix A). We hope to maintain 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.

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 requires users to understand how LLMs generate these specifications, emphasizing the need for explainable planning to yield robust, transparent, and correctable outputs (Zuo et al., 2024). Corrective feedback loops notably improve failure handling, such as resolving action precondition errors (Raman et al., 2024), or re-planning in case of unexpected failures during plan execution (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 (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 et al. (2024) and Smirnov et al. (2024). A potential 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) that are easier for LLMs to process before converting to PDDL (Agarwal and Sreepathy, 2024; Smirnov et al., 2024) can also enhance accuracy.

Revisiting RQ2: LLMs have demonstrated that they are significantly sensitive to prompting—raising questions about whether they are better off functioning as *translators* or *generators*. Liu et al. (2023a) demonstrate that highly explicit descriptions improve translation accuracy, while Gestrin et al. (2024) leverage minimal descriptions, relying on LLMs’ internal world knowledge to enrich outputs; however, this excess freedom often leads to inconsistent or inexecutable domain models. Huang et al. (2024b); Liu et al. (2023a); Guan et al. (2023) recognize that specifying a precise predicate set in NL 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 underscores the importance of standardizing prompt granularity for initial generation and iterative feedback (Liu et al., 2024b). Nabizada et al. (2024) provide a promising, organized, and standardized paradigm for automatically generating PDDL descriptions that can be applied to LLMs.

6 Conclusion

Extracting planning models has long been recognized as a major barrier to the widespread adoption of planning technologies (Vallati and Kitchin, 2020; Hendler et al., 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.

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 temporal 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

- Sudhir Agarwal and Anu Sreepathy. 2024. [TIC: translate-infer-compile for accurate 'text to plan' using llms and logical intermediate representations](#). *CoRR*, abs/2402.06608.
- Sudhir Agarwal, Anu Sreepathy, David H. Alonso, and Prarit Lamba. 2024. [Llm+reasoning+planning for supporting incomplete user queries in presence of apis](#). *CoRR*, abs/2405.12433.
- Daman Arora and Subbarao Kambhampati. 2023. [Learning and leveraging verifiers to improve planning capabilities of pre-trained language models](#). *CoRR*, abs/2305.17077.
- Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2024. [Predicate invention from pixels via pretrained vision-language models](#). *arXiv preprint arXiv:2501.00296*.
- Gregor Behnke and Pascal Bercher. 2024. [Envisioning a domain learning track for the ipc](#).
- Yoshua Bengio. 2020. [Deep learning for system 2 processing](#). *AAAI 2020*.
- Timo Birr, Christoph Pohl, Abdelrahman Younes, and Tamim Asfour. 2024. [Autogpt+p: Affordance-based task planning using large language models](#). In *Robotics: Science and Systems XX*, RSS2024. Robotics: Science and Systems Foundation.
- Bernd Bohnet, Azade Nova, Aaron T. Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. [Exploring and benchmarking the planning capabilities of large language models](#). *CoRR*, abs/2406.13094.
- Owen Burns, Dana Hughes, and Katia P. Sycara. 2024. [Plancritic: Formal planning with human feedback](#). *CoRR*, abs/2412.00300.
- Turgay Caglar, Sirine Belhaj, Tathagata Chakraborti, Michael Katz, and Sarath Sreedharan. 2024. [Can llms fix issues with reasoning models? towards more likely models for AI planning](#). In *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*, pages 20061–20069. AAAI Press.
- Daniel Cao, Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. 2024. [Automating thought of search: A journey towards soundness and completeness](#). *CoRR*, abs/2408.11326.
- Georgia Chalvatzaki, Ali Younes, Daljeet Nandha, An T. Le, Leonardo F. R. Ribeiro, and Iryna Gurevych. 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.
- Haonan Chang, Kai Gao, Kowndinya Boyalakuntla, Alex Lee, Baichuan Huang, Harish Udhaya Kumar, Jinjin Yu, and Abdeslam Boularias. 2023. [LGM-CTS: language-guided monte-carlo tree search for executable semantic object rearrangement](#). *CoRR*, abs/2309.15821.
- Guanqi Chen, Lei Yang, Ruixing Jia, Zhe Hu, Yizhou Chen, Wei Zhang, Wenping Wang, and Jia Pan. 2024. [Language-augmented symbolic planner for open-world task planning](#). *CoRR*, abs/2407.09792.
- Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2023a. [Autotamp: Autoregressive task and motion planning with llms as translators and checkers](#). *CoRR*, abs/2306.06531.

- Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. 2023b. [NL2TL: transforming natural languages to temporal logics using large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15880–15903. Association for Computational Linguistics.
- Katherine M Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B Tenenbaum. 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*.
- Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. 2023. [nl2spec: Interactively translating unstructured natural language to temporal logics with large language models](#). *CoRR*, abs/2303.04864.
- Luis Miguel Vieira da Silva, Aljosha Köcher, Felix Gehlhoff, and Alexander Fay. 2024. [Toward a method to generate capability ontologies from natural language descriptions](#). *CoRR*, abs/2406.07962.
- Gautier Dagan, Frank Keller, and Alex Lascarides. 2023. [Dynamic planning with a LLM](#). *CoRR*, abs/2308.06391.
- Kahneman Daniel. 2017. *Thinking, fast and slow*. Macmillan.
- Tomás de la Rosa, Sriram Gopalakrishnan, Alberto Pozanco, Zhen Zeng, and Daniel Borrajo. 2024. [TRIP-PAL: travel planning with guarantees by combining large language models and automated planners](#). *CoRR*, abs/2406.10196.
- Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2023a. [Integrating action knowledge and llms for task planning and situation handling in open worlds](#). *CoRR*, abs/2305.17590.
- Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. 2023b. [Task and motion planning with large language models for object rearrangement](#). *CoRR*, abs/2303.06247.
- Wenfeng Feng, Hankz Hankui Zhuo, and Subbarao Kambhampati. 2018. [Extracting action sequences from texts based on deep reinforcement learning](#). *CoRR*, abs/1803.02632.
- Morgan Fine-Morris, Vincent Hsiao, Leslie N Smith, Laura M Hiatt, and Mark Roberts. 2024. [Leveraging llms for generating document-informed hierarchical planning models: A proposal](#). In *AAAI 2025 Workshop LM4Plan*.
- Elliot Gestrin, Marco Kuhlmann, and Jendrik Seipp. 2024. [NI2plan: Robust llm-driven planning from minimal text descriptions](#). *CoRR*, abs/2405.04215.
- Alba Gragera and Alberto Pozanco. 2023. [Exploring the limitations of using large language models to fix planning tasks](#). In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.
- Sachin Grover and Shiwali Mohan. 2024. [A demonstration of natural language understanding in embodied planning agents](#). In *ICAPS 2024 System’s Demonstration track*.
- Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. [Leveraging pre-trained large language models to construct and utilize world models for model-based task planning](#). In *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*.
- Atharva Gundawar, Mudit Verma, Lin Guan, Karthik Valmeekam, Siddhant Bhambri, and Subbarao Kambhampati. 2024. [Robust planning with llm-modulo framework: Case study in travel planning](#). *CoRR*, abs/2405.20625.
- Weihang Guo, Zachary K. Kingston, and Lydia E. Kavvaki. 2024. [Castl: Constraints as specifications through LLM translation for long-horizon task and motion planning](#). *CoRR*, abs/2410.22225.
- Abhinav Gupta, Alexei A Efros, and Martial Hebert. 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*, pages 482–496. Springer.
- Muzhi Han, Yifeng Zhu, Song-Chun Zhu, Ying Nian Wu, and Yuke Zhu. 2024. [Interpret: Interactive predicate learning from language feedback for generalizable task planning](#). *CoRR*, abs/2405.19758.
- Yilun Hao, Yang Zhang, and Chuchu Fan. 2024. [Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming](#). *CoRR*, abs/2410.12112.
- Thomas Hayton, Julie Porteous, João F. Ferreira, and Alan Lindsay. 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*, pages 1709–1716. AAAI Press.
- James A. Hendler, Austin Tate, and Mark Drummond. 1990. [AI planning: Systems and techniques](#). *AI Mag.*, 11(2):61–77.
- Eran Hirsch, Guy Uziel, and Ateret Anaby-Tavor. 2024. [What’s the plan? evaluating and developing planning-aware techniques for llms](#). *CoRR*, abs/2402.11489.

- Richard Howey, Derek Long, and Maria Fox. 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*, pages 294–301. IEEE Computer Society.
- Mengkang Hu, Tianxing Chen, Yude Zou, Yuheng Lei, Qiguang Chen, Ming Li, Yao Mu, Hongyuan Zhang, Wenqi Shao, and Ping Luo. 2025. [Text2world: Benchmarking large language models for symbolic world model generation](#).
- Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, Saravan Rajmohan, and Dongmei Zhang. 2024. [Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation](#). *CoRR*, abs/2408.00764.
- Cassie Huang and Li Zhang. 2024. [On the limit of language models as planning formalizers](#). *CoRR*, abs/2412.09879.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 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.
- Ruoyun Huang, Yixin Chen, and Weixiong Zhang. 2014. [SAS+ planning as satisfiability](#). *CoRR*, abs/1401.4598.
- Sukai Huang, Nir Lipovetzky, and Trevor Cohn. 2024b. [Planning in the dark: Llm-symbolic planning pipeline without experts](#). *arXiv preprint arXiv:2409.15915*.
- Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024c. [Understanding the planning of LLM agents: A survey](#). *CoRR*, abs/2402.02716.
- Adam Ishay and Joohyung Lee. 2025. [Llm+ al: Bridging large language models and action languages for complex reasoning about actions](#). *arXiv preprint arXiv:2501.00830*.
- Silvia Izquierdo-Badiola, Gerard Canal, Carlos Rizzo, and Guillem Alenyà. 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*, pages 17344–17350. IEEE.
- Frank Joubin, Antonello Ceravola, Pavel Smirnov, Felix Ocker, Joerg Deigmoeller, Anna Belardinelli, Chao Wang, Stephan Hasler, Daniel Tanneberg, and Michael Gienger. 2023. [Copal: Corrective planning of robot actions with large language models](#). *CoRR*, abs/2310.07263.
- Elias Helle Kalland. 2024. [Enabling semantic reasoning in robots through natural language processing](#). Master’s thesis, NTNU.
- Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. [Llms can’t plan, but can help planning in llm-modulo frameworks](#). *CoRR*, abs/2402.01817.
- Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi Araghi. 2024. [Thought of search: Planning with language models through the lens of efficiency](#). *Advances in Neural Information Processing Systems*, 37:138491–138568.
- Jack Kelly, Alex Calderwood, Noah Wardrip-Fruin, and Michael Mateas. 2023. [There and back again: Extracting formal domains for controllable neurosymbolic story authoring](#). In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, October 08-12, 2023, Salt Lake City, UT, USA*, pages 64–74. AAAI Press.
- Vedant Khandelwal, Amit P. Sheth, and Forest Agostinelli. 2024. [PddlFuse: A tool for generating diverse planning domains](#). *CoRR*, abs/2411.19886.
- Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. 2024. [Acpbench: Reasoning about action, change, and planning](#). *CoRR*, abs/2410.05669.
- Minseo Kwon, Yaesol Kim, and Young J. Kim. 2024. [Fast and accurate task planning using neuro-symbolic language models and multi-level goal decomposition](#). *CoRR*, abs/2409.19250.
- Jacqueline Lee, Michelle Cantu, Joel Korb, Eva Meth, John D Griffith, Joanna Korman, Anna Yuen, Peter Schwartz, and Abigail S Gertner. 2024. [Planning ai assistant for emergency decision-making \(planaid\): Framing planning problems and assessing plans with large language models](#). In *AAAI 2025 Workshop LM4Plan*.
- Haoming Li, Zhaoliang Chen, Jonathan Zhang, and Fei Liu. 2024a. [LASP: surveying the state-of-the-art in large language model-assisted AI planning](#). *CoRR*, abs/2409.01806.
- Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kurnakov, Fedor Zhdanov, Manuel Romero, Tony Lee,

- Nadav Timor, Jennifer Ding, Claire Schlesinger, Hai-ley Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. [Starcoder: may the source be with you!](#) *CoRR*, abs/2305.06161.
- Siyan Li, Zhe Ma, Feifan Liu, Jiani Lu, Qinqin Xiao, Kewu Sun, Lingfei Cui, Xirui Yang, Peng Liu, and Xun Wang. 2024b. [Safe planner: Empowering safety awareness in large pre-trained models for robot task planning.](#) *CoRR*, abs/2411.06920.
- Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. [Text2motion: from natural language instructions to feasible plans.](#) *Auton. Robots*, 47(8):1345–1365.
- Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. [LLM+P: empowering large language models with optimal planning proficiency.](#) *CoRR*, abs/2304.11477.
- Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. 2023b. [Lang2ttl: Translating natural language commands to temporal robot task specification.](#) *CoRR*, abs/2302.11649.
- Weiyu Liu, Neil Nie, Jiayuan Mao, Ruohan Zhang, and Jiajun Wu. 2024a. Learning compositional behaviors from demonstration and language. In *8th Annual Conference on Robot Learning*.
- Yuchen Liu, Luigi Palmieri, Sebastian Koch, Ilche Georgievski, and Marco Aiello. 2024b. [DELTA: decomposed efficient long-term robot task planning using large language models.](#) *CoRR*, abs/2404.03275.
- Yuchen Liu, Luigi Palmieri, Sebastian Koch, Ilche Georgievski, and Marco Aiello. 2024c. [Towards human awareness in robot task planning with large language models.](#) *CoRR*, abs/2404.11267.
- Xusheng Luo, Shaojun Xu, and Changliu Liu. 2023. Obtaining hierarchy from human instructions: an llms-based approach. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*.
- Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. [Faithful chain-of-thought reasoning.](#) *CoRR*, abs/2301.13379.
- Sadeqh Mahdavi, Raquel Aoki, Keyi Tang, and Yan-shuai Cao. 2024. [Leveraging environment interaction for automated PDDL generation and planning with large language models.](#) *CoRR*, abs/2407.12979.
- Kumar Manas, Stefan Zwicklbauer, and Adrian Paschke. 2024. [Cot-tl: Low-resource temporal knowledge representation of planning instructions using chain-of-thought reasoning.](#) *CoRR*, abs/2410.16207.
- Yuchen Mao, Tianci Zhang, Xu Cao, Zhongyao Chen, Xinkai Liang, Bochen Xu, and Hao Fang. 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*, pages 469–474. IEEE.
- D. McDermott, A. Howe M. Ghallab, C. Knoblock, M. Veloso A. Ram, D. Weld, and D. Wilkins. 1998. [Pddl-the planning domain definition language.](#)
- Israel Puerta Merino and Jordi Sabater-Mir. 2024. [LLM reasoner and automated planner: A new NPC approach.](#) In *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*, pages 244–247. IOS Press.
- Shivam Miglani and Neil Yorke-Smith. 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.
- Ida Momennejad, Hosein Hasanbeig, Felipe Vieira Frutteri, Hiteshi Sharma, Nebojsa Jovic, Hamid Palangi, Robert Osazuwa Ness, and Jonathan Larson. 2023. [Evaluating cognitive maps and planning in large language models with cogeval.](#) In *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*.
- William Murphy, Nikolaus Holzer, Nathan Koenig, Leyi Cui, Raven Rothkopf, Feitong Qiao, and Mark Santolucito. 2024. [Guiding LLM temporal logic generation with explicit separation of data and control.](#) *CoRR*, abs/2406.07400.
- Hamied Nabizada, Tom Jeleniewski, Felix Gehlhoff, and Alexander Fay. 2024. [Model-based workflow for the automated generation of PDDL descriptions.](#) *CoRR*, abs/2408.08145.
- Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 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.
- Tim Oates, Ron Alford, Shawn Johnson, and Cory Hall. 2024. Using large language models to extract planning knowledge from common vulnerabilities and exposures.
- James T. Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. 2024.

- Large language models as planning domain generators. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, pages 423–431. AAAI Press.
- Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Lior Horesh, Biplav Srivastava, Francesco Fabiano, and Andrea Loreggia. 2022. **Plansformer: Generating symbolic plans using transformers**. *CoRR*, abs/2212.08681.
- Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Biplav Srivastava, Lior Horesh, Francesco Fabiano, and Andrea Loreggia. 2023. **Understanding the capabilities of large language models for automated planning**. *CoRR*, abs/2305.16151.
- Vishal Pallagani, Bharath C. Muppasani, Kaushik Roy, Francesco Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, Lior Horesh, and Amit P. Sheth. 2024. **On the prospects of incorporating large language models (llms) in automated planning and scheduling (APS)**. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, pages 432–444. AAAI Press.
- Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. **Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning**. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3806–3824. Association for Computational Linguistics.
- Kshitij Patil. 2024. **Llms for ai planning: a study on error detection and correction in pddl domain models**. Master’s thesis.
- David Paulius, Alejandro Agostini, Benedict Quartey, and George Konidaris. 2024. **Bootstrapping object-level planning with large language models**. *arXiv preprint arXiv:2409.12262*.
- Israel Puerta-Merino, Carlos Núñez-Molina, Pablo Mesejo, and Juan Fernández-Olivares. 2025. **A roadmap to guide the integration of llms in hierarchical planning**. *arXiv preprint arXiv:2501.08068*.
- Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Ray Mooney, Stefanie Tellex, and David Paulius. 2024. **Cape: Corrective actions from precondition errors using large language models**.
- Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. 2022. **Planning with large language models via corrective re-prompting**. *CoRR*, abs/2211.09935.
- Bernardino Romera-Paredes, Mohammadamin Barekatin, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. **Mathematical discoveries from program search with large language models**. *Nat.*, 625(7995):468–475.
- Stuart Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson.
- Md Sadman Sakib and Yu Sun. 2024. **Consolidating trees of robotic plans generated using large language models to improve reliability**. *CoRR*, abs/2401.07868.
- Bilgehan Sel, Ruoxi Jia, and Ming Jin. 2025. **Llms can plan only if we tell them**. *arXiv preprint arXiv:2501.13545*.
- Naman Shah. 2024. **Autonomously learning world-model representations for efficient robot planning**. Technical report, Arizona State University.
- Keisuke Shirai, Cristian C. Beltran-Hernandez, Masashi Hamaya, Atsushi Hashimoto, Shohei Tanaka, Kento Kawaharazuka, Kazutoshi Tanaka, Yoshitaka Ushiku, and Shinsuke Mori. 2023. **Vision-language interpreter for robot task planning**. *CoRR*, abs/2311.00967.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 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.
- Kelsey Sikes, Morgan Fine-Morris, Sarath Sreedharan, and Mark Roberts. 2024a. **Traversing the linguistic divide: Aligning semantically equivalent fluents through model refinement**.
- Kelsey Sikes, Morgan Fine-Morris, Sarath Sreedharan, Leslie N Smith, and Mark Roberts. 2024b. **Creating pddl models from javascript using llms: Preliminary results**. In *AAAI 2025 Workshop LM4Plan*.
- Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2022. **Pddl planning with pretrained large language models**. In *NeurIPS 2022 foundation models for decision making workshop*.
- Nisha Simon and Christian Muise. 2021. **A natural language model for generating pddl**. In *ICAPS KEPS workshop*.
- Ishika Singh, David Traum, and Jesse Thomason. 2024a. **Twostep: Multi-agent task planning using classical planners and large language models**. *CoRR*, abs/2403.17246.
- Shivam Singh, Karthik Swaminathan, Raghav Arora, Ramandeep Singh, Ahana Datta, Dipanjan Das, Snehasis Banerjee, Mohan Sridharan, and K. Madhava Krishna. 2024b. **Anticipate & collab: Data-driven task anticipation and knowledge-driven planning for human-robot collaboration**. *CoRR*, abs/2404.03587.

- Vishesh Sinha. 2024. Leveraging llms for htn domain model generation via prompt engineering. Master’s thesis.
- Pavel Smirnov, Frank Joublin, Antonello Ceravola, and Michael Gienger. 2024. [Generating consistent PDDL domains with large language models](#). *CoRR*, abs/2404.07751.
- Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. [GPT-4 doesn’t know it’s wrong: An analysis of iterative prompting for reasoning problems](#). *CoRR*, abs/2310.12397.
- Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. [Chain of thoughtlessness: An analysis of cot in planning](#). *CoRR*, abs/2405.04776.
- Katharina Stein and Alexander Koller. 2023. [Autoplanbench: : Automatically generating benchmarks for LLM planners from PDDL](#). *CoRR*, abs/2311.09830.
- Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. [Solving olympiad geometry without human demonstrations](#). *Nat.*, 625(7995):476–482.
- Alexander Tuisov, Yonatan Vernik, and Alexander Shleyfman. 2025. Llm-generated heuristics for ai planning: Do we even need domain-independence anymore? *arXiv preprint arXiv:2501.18784*.
- Mauro Vallati and Diane E. Kitchin, editors. 2020. *Knowledge Engineering Tools and Techniques for AI Planning*. Springer.
- Karthik Valmeekam, Matthew Marquez, Alberto Olmo Hernandez, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. [Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change](#). In *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*.
- Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023b. [Can large language models really improve by self-critiquing their own plans?](#) *CoRR*, abs/2310.08118.
- Karthik Valmeekam, Kaya Stechly, Atharva Gundawar, and Subbarao Kambhampati. 2024a. [Planning in strawberry fields: Evaluating and improving the planning and scheduling capabilities of lrm o1](#). *arXiv preprint arXiv:2410.02162*.
- Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024b. [Llms still can’t plan; can lrms? a preliminary evaluation of openai’s o1 on planbench](#). *arXiv preprint arXiv:2409.13373*.
- Lev Semyonovich Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Harvard UP.
- Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. 2024. [Planning in natural language improves LLM search for code generation](#). *CoRR*, abs/2409.03733.
- Fangyuan Wang, Shipeng Lyu, Peng Zhou, Anqing Duan, Guodong Guo, and David Navarro-Alarcon. 2025. [Instruction-augmented long-horizon planning: Embedding grounding mechanisms in embodied mobile manipulation](#).
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. [Emergent abilities of large language models](#). *CoRR*, abs/2206.07682.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. [Chain-of-thought prompting elicits reasoning in large language models](#). In *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*.
- Lionel Wong, Jiayuan Mao, Pratyusha Sharma, Zachary S. Siegel, Jiahai Feng, Noa Korneev, Joshua B. Tenenbaum, and Jacob Andreas. 2023. [Learning adaptive planning representations with natural language guidance](#). *CoRR*, abs/2312.08566.
- Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 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.
- Kaige Xie, Ian Yang, John Gunerli, and Mark Riedl. 2024b. [Making large language models into world models with precondition and effect knowledge](#). *arXiv preprint arXiv:2409.12278*.
- Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. 2023. [Translating natural language to planning goals with large-language models](#). *CoRR*, abs/2302.05128.
- Ruolin Ye, Yifei Hu, Yuhan Anjelica Bian, Luke Kulm, and Tapomayukh Bhattacharjee. 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*, pages 9540–9547. IEEE.
- Lance Ying, Katherine M. Collins, Megan Wei, Cedegao E. Zhang, Tan Zhi-Xuan, Adrian Weller, Joshua B. Tenenbaum, and Lionel Wong. 2023. [The neuro-symbolic inverse planning engine \(NIPE\): modeling probabilistic social inferences from linguistic inputs](#). *CoRR*, abs/2306.14325.

- Li Zhang. 2024. [Structured event reasoning with large language models](#). *CoRR*, abs/2408.16098.
- Li Zhang, Peter Jansen, Tianyi Zhang, Peter Clark, Chris Callison-Burch, and Niket Tandon. 2024a. [PDDLEGO: iterative planning in textual environments](#). *CoRR*, abs/2405.19793.
- Tianyi Zhang, Li Zhang, Zhaoyi Hou, Ziyu Wang, Yuling Gu, Peter Clark, Chris Callison-Burch, and Niket Tandon. 2024b. [PROC2PDDL: open-domain planning representations from texts](#). *CoRR*, abs/2403.00092.
- Xiaohan Zhang, Zainab Altaweel, Yohei Hayamizu, Yan Ding, Saeid Amiri, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2024c. [DKPROMPT: domain knowledge prompting vision-language models for open-world planning](#). *CoRR*, abs/2406.17659.
- Xiaopan Zhang, Hao Qin, Fuquan Wang, Yue Dong, and Jiachen Li. 2024d. [Lamma-p: Generalizable multi-agent long-horizon task allocation and planning with lm-driven PDDL planner](#). *CoRR*, abs/2409.20560.
- Zhigen Zhao, Shuo Cheng, Yan Ding, Ziyi Zhou, Shiqi Zhang, Danfei Xu, and Ye Zhao. 2024. [A survey of optimization-based task and motion planning: From classical to learning approaches](#). *CoRR*, abs/2404.02817.
- Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2024. [NATURAL PLAN: benchmarking llms on natural language planning](#). *CoRR*, abs/2406.04520.
- Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2023. [ISR-LLM: iterative self-refined large language model for long-horizon sequential task planning](#). *CoRR*, abs/2308.13724.
- Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael L. Littman, and Stephen H. Bach. 2024. [Planetarium: A rigorous benchmark for translating text to structured planning languages](#). *CoRR*, abs/2407.03321.

A Paper Overview

Below is a quick summary of core model generation frameworks in this survey. Papers are sorted by planning specification coverage, followed by publication year, and then alphabetically by author.

WORKS		PDDL SPEC.				GUIDANCE		
Paper	Framework	Init.	Goal	Preds.	Act.	Prompt	Feedback	Human Int.
*(Gestrin et al., 2024)	NL2PLAN	✓	✓	✓	✓	Heavy (Few-shot+CoT)	Custom validator + LLM	Optional
(Liu et al., 2024b)	DELTA	✓	✓	✓	✓	Medium (Few-shot)	None	None
(Mahdavi et al., 2024)	LLM+EW	✓	✓	✓	✓	Light	Env. (EW Metric) + LLM	None
(Smirnov et al., 2024)	—	✓	✓	✓	✓	Light (JSON interm.)	Custom validator + LLM	Optional
(Sakib and Sun, 2024)	—	✓	✓	✓	✓	Medium (Few-shot)	None	None
(Ye et al., 2024)	MORPHeus	✓	✓	✓	✓	Heavy (Few-shot+CoT)	NL Human feedback	True
(Kelly et al., 2023)	TABA	✓	✓	✓	✓	Light (One-shot)	Glaive validator + LLM	Optional
(Ying et al., 2023)	NIPE	✓	✓	✓	✓	Medium (Few-shot)	None	None
(Zhou et al., 2023)	ISR-LLM	✓	✓	✓	✓	Medium (Few-shot)	None	None
(Han et al., 2024)	InterPret	×	✓	✓	✓	Light (CoT)	NL Human feedback	True
(Liu et al., 2024c)	—	×	×	✓	✓	Medium (unknown)	Scene graph	True
*(Hu et al., 2025)	TEXT2WORLD	×	×	✓	✓	Heavy (Few-shot+Exp.)	Tarski validator + LLM	None
(Sinha, 2024)	—	×	×	✓	✓	Heavy (Few-shot+Exp.)	None	None
*(Sikes et al., 2024b)	—	×	×	✓	✓	Medium (Few-shot)	None	None
*(Guan et al., 2023)	LLM+DM	×	×	✓	✓	Heavy (Few-shot+CoT+Exp.)	Custom validator + LLM	None
(Chen et al., 2024)	LASP	×	×	×	✓	Medium (High Exp.)	LLM	None
(Huang et al., 2024b)	—	×	×	×	✓	Medium (Few-shot)	VSCoDe-PDDL validator	None
(Liu et al., 2024a)	BLADE	×	×	×	✓	Heavy	Human demonstration	True
(Oates et al., 2024)	CLLaMP	×	×	×	✓	Medium (Few-shot)	None	None
(Oswald et al., 2024)	NL2PDDL	×	×	×	✓	Medium (Few-shot)	None	None
(Wong et al., 2023)	ADA	×	×	×	✓	Medium (Few-shot+Exp.)	None	None
*(Zhang et al., 2024b)	PROC2PDDL	×	×	×	✓	Heavy (Few-shot)	None	None
(Athalye et al., 2024)	pix2pred	×	×	✓	×	Light	Candidate filtering	None
(Wang et al., 2025)	IALP	✓	✓	×	×	Medium (Few-shot+VLM)	VLM	None
(Agarwal and Sreepathy, 2024)	TIC	✓	✓	×	×	Light (Zero/Few-shot)	None	None
(de la Rosa et al., 2024)	TRIP-PAL	✓	✓	×	×	Light	None	None
(Guo et al., 2024)	CaSTL	✓	✓	×	×	Medium (Exp.)	Python validator + LLM	None
(Kalland, 2024)	SemReBot2	✓	✓	×	×	Medium (Few-shot+Exp.)	None	None
(Kwon et al., 2024)	—	✓	✓	×	×	Medium (Few-shot)	None	None
(Lee et al., 2024)	PlanAID	✓	✓	×	×	Light (Single-shot)	Human deviation	True
(Zhang et al., 2024a)	PDDLEGO	✓	✓	×	×	Medium (Few-shot+Exp.)	LLM (PDDL-edit)	None
(Zhang et al., 2024d)	LaMMA-P	✓	✓	×	×	Medium (Few-shot)	FastDownward	None
*(Liu et al., 2023a)	LLM+P	✓	✓	×	×	Medium (Few-shot)	None	None
(Shirai et al., 2023)	ViLaIn	✓	✓	×	×	Medium (Few-shot+VLM)	FastDownward	None
(Birr et al., 2024)	Auto-GPT+P	×	✓	×	×	Light	PDDL validator + Prolog	True
(Grover and Mohan, 2024)	—	×	✓	×	×	Medium (Few-shot)	AI2Thor Env.	None
(Izquierdo-Badiola et al., 2024)	PlanCollabNL	×	✓	×	×	Light	LLM	None
(Li et al., 2024b)	SafePlanner	×	✓	×	×	Medium (Exp.)	None	None
(Paulius et al., 2024)	OLP	×	✓	×	×	Light	None	None
(Singh et al., 2024b)	DaTAPlan	×	✓	×	×	Medium (Few-shot/CoT+Exp.)	Human deviation	True
(Singh et al., 2024a)	TwoStep	×	✓	×	×	Medium (Few-shot)	LLM	None
(Dagan et al., 2023)	LLM+DP	×	✓	×	×	Medium (Few-shot+Exp.)	Alfworld Env.	None
(Lyu et al., 2023)	Faithful CoT	×	✓	×	×	Light (CoT)	None	None
(Xie et al., 2023)	—	×	✓	×	×	Medium (Few-shot)	None	None
*(Collins et al., 2022)	P+S	×	✓	×	×	Medium (Few-shot)	None	None

Figure 4: *Exp.* = Explicit PDDL info. *Feedback/Human Intervention* provided at the level of the LLM-generated PDDL spec. itself. *Papers reconstructed by L2P.

B Additional Information on PDDL

B.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 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. Instead of writing executable logic, LLMs assist in mapping natural language descriptions to structured symbolic representations.

B.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 $\mathbb{D}\mathbb{F}$:

```
(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 $\mathbb{P}\mathbb{F}$:

```
(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
```

C (L2P) Framework

C.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 7, which shows how LLM feedback can be used to refine a PDDL problem specification.

C.2 Paper Reconstructions

L2P can recreate and encompass previous frameworks for converting natural language to PDDL, serving as a comprehensive foundation that integrates past approaches. L2P 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; action and predicate output can be found in Figure 8.

C.3 L2P Usage Example

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

```
1 import os
2 from l2p.llm.openai import OPENAI
3 from l2p.utils import load_file
4 from l2p.domain_builder import DomainBuilder
5
6 domain_builder = DomainBuilder()
7
8 api_key = os.environ.get('OPENAI_API_KEY')
9 llm = OPENAI(model="gpt-4o-mini", api_key=api_key)
10
11 # retrieve prompt information
12 base_path='tests/usage/prompts/domain/'
13 domain_desc = load_file(f'{base_path}blocksworld_domain.txt')
14 predicates_prompt = load_file(f'{base_path}formalize_predicates.txt')
15 types = load_file(f'{base_path}types.json')
16 action = load_file(f'{base_path}action.json')
17
18 # extract predicates via LLM
19 predicates, llm_output, validation_info = domain_builder.formalize_predicates(
20     model=llm,
21     domain_desc=domain_desc,
22     prompt_template=predicates_prompt,
23     types=types
24 )
25
26 # format key info into PDDL strings
27 predicate_str = "\n".join([pred["raw"].replace(":", " ; ") for pred in predicates])
28
29 print(f"###OUTPUT\n{predicate_str}")
30 -----
31
32 ### OUTPUT
33 - (holding ?a - arm ?b - block) ; true if the arm ?a is currently holding the block ?b
34 - (on_table ?b - block) ; true if the block ?b is on the table
35 - (clear ?b - block) ; true if the block ?b is clear (no block on top of it)
36 - (on_top ?b1 - block ?b2 - block) ; true if the block ?b1 is on top of the block ?b2
```

Figure 5: L2P usage - generating simple PDDL predicates

```

1  import os
2  from l2p.utils.pddl_types import Predicate
3  from l2p.task_builder import TaskBuilder
4
5  task_builder = TaskBuilder() # initialize task builder class
6  api_key = os.environ.get('OPENAI_API_KEY')
7  llm = OPENAI(model="gpt-4o-mini", api_key=api_key)
8
9  # load in assumptions
10 problem_desc = load_file(r'tests/usage/prompts/problem/blocksworld_problem.txt')
11 task_prompt = load_file(r'tests/usage/prompts/problem/formalize_task.txt')
12 types = load_file(r'tests/usage/prompts/domain/types.json')
13 predicates_json = load_file(r'tests/usage/prompts/domain/predicates.json')
14 predicates: List[Predicate] = [Predicate(**item) for item in predicates_json]
15
16 # extract PDDL task specifications via LLM
17 objects, init, goal, llm_response, validation_info = task_builder.formalize_task(
18     model=llm,
19     problem_desc=problem_desc,
20     prompt_template=task_prompt,
21     types=types,
22     predicates=predicates
23 )
24
25 # generate task file
26 pddl_problem = task_builder.generate_task(
27     domain_name="blocksworld",
28     problem_name="blocksworld_problem",
29     objects=objects,
30     initial=init,
31     goal=goal)
32
33 print(f"### LLM OUTPUT:\n {pddl_problem}")
34 -----
35 ### LLM OUTPUT:
36 (define
37   (problem blocksworld_problem)
38   (:domain blocksworld)
39
40   (:objects
41     arm1 - arm
42     blue_block - block
43     red_block - block
44     yellow_block - block
45     green_block - block
46   )
47
48   (:init
49     (on_top blue_block red_block)
50     (on_top red_block yellow_block)
51     (on_table yellow_block)
52     (on_table green_block)
53     (clear blue_block)
54     (clear green_block)
55   )
56
57   (:goal
58     (and
59       (on_top red_block green_block)
60     )
61   )
62 )

```

Figure 6: L2P usage - generating simple PDDL task specification

```

1  import os
2  from l2p.feedback_builder import FeedbackBuilder
3
4  feedback_builder = FeedbackBuilder()
5  api_key = os.environ.get('OPENAI_API_KEY')
6  llm = OPENAI(model="gpt-4o-mini", api_key=api_key)
7
8  problem_desc = load_file(r'tests/usage/prompts/problem/blocksworld_problem.txt')
9  types = load_file(r'tests/usage/prompts/domain/types.json')
10 feedback_template = load_file(r'tests/usage/prompts/problem/feedback.txt')
11 predicates_json = load_file(r'tests/usage/prompts/domain/predicates.json')
12 predicates: list[Predicate] = [Predicate(**item) for item in predicates_json]
13 llm_response = load_file(r'tests/usage/prompts/domain/llm_output_task.txt')
14
15 fb_pass, feedback_response = feedback_builder.task_feedback(
16     model=llm,
17     problem_desc=problem_desc,
18     llm_output=llm_response,
19     feedback_template=feedback_template,
20     feedback_type="llm",
21     predicates=predicates,
22     types=types)
23
24 print("[FEEDBACK]\n", feedback_response)
25 -----
26 [FEEDBACK]
27 ### JUDGMENT
28 ---
29 My feedback on the provided PDDL problem file is as follows:
30
31 1. Are any necessary objects missing?
32 All necessary blocks are included. Therefore: No.
33
34 2. Are any unnecessary objects included?
35 All objects are relevant to the problem. Hence: No.
36
37 3. Are any objects defined with the wrong type?
38 All objects are correctly defined as "object". Therefore: No.
39
40 4. Are any unnecessary or incorrect predicates declared?
41 All predicates used are relevant and correctly applied. Thus: No.
42
43 5. Are any needed or expected predicates missing from the initial state?
44 The initial state is missing the predicate for the red block being not clear, as it is covered
45 by the blue block. This should be added:
46 - (clear red_block) should be false, but it is not explicitly stated. Hence: Yes.
47
48 6. Is anything missing from the goal state?
49 The goal state is correctly defined as having the red block on top of the green block. So: No.
50
51 7. Is anything unnecessary included in the goal description?
52 The goal description is concise and only includes what is necessary. Therefore: No.
53
54 8. Should any predicate be used in a symmetrical manner?
55 The predicates used do not require symmetry in this context. Hence: No.
56
57 In summary, the main issue is the missing predicate regarding the clarity of the red block. It
58 should be explicitly stated that the red block is not clear due to the blue block being on
59 top of it.
60 - (clear red_block) should be false, or alternatively, you can add:
61 - (not (clear red_block)) to indicate that the red block is not clear.
62 ---

```

Figure 7: 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'}

## ACTIONS
{'name': 'load_truck', 'parameters': OrderedDict([('p', 'package'), ('t', 'truck'), ('l', 'location')]),
 'preconditions': '(and\n (truck-at ?t ?l)\n (package-at ?p ?l)\n (truck-has-space ?t)\n)',
 'effects': '(and\n (not (package-at ?p ?l))\n (truck-holding ?t ?p)\n)'}
{'name': 'unload_truck', 'parameters': OrderedDict([('p', 'package'), ('t', 'truck'), ('l', 'location')]),
 'preconditions': '(and\n (truck-at ?t ?l)\n (truck-holding ?t ?p)\n)',
 'effects': '(and\n (not (truck-holding ?t ?p))\n (package-at ?p ?l)\n)'}
{'name': 'load_airplane', 'parameters': OrderedDict([('p', 'package'), ('a', 'plane')]),
 'preconditions': '(and\n (package-at ?p ?l)\n (plane-at ?a ?l)\n)',
 'effects': '(and\n (not (package-at ?p ?l))\n (plane-holding ?a ?p)\n)'}
{'name': 'unload_airplane', 'parameters': OrderedDict([('p', 'package'), ('a', 'plane'), ('l', 'location')]),
 'preconditions': '(and\n (plane-at ?a ?l)\n (plane-holding ?a ?p)\n)',
 'effects': '(and\n (not (plane-holding ?a ?p))\n (package-at ?p ?l)\n)'}
{'name': 'drive_truck',
 'parameters': OrderedDict([('t', 'truck'), ('l1', 'location'), ('l2', 'location'), ('c', 'city')]),
 'preconditions': '(and\n (truck-at ?t ?l1)\n (connected-locations ?l1 ?l2 ?c)\n)',
 'effects': '(and\n (not (truck-at ?t ?l1))\n (truck-at ?t ?l2)\n)'}
{'name': 'fly_airplane',
 'parameters': OrderedDict([('a', 'plane'), ('l1', 'location'), ('l2', 'location'), ('c', 'city')]),
 'preconditions': '(and\n (plane-at ?a ?l1)\n (connected-locations ?l1 ?l2 ?c)\n)',
 'effects': '(and\n (not (plane-at ?a ?l1))\n (plane-at ?a ?l2)\n)'}

```

Figure 8: L2P formatted predicate and actions outputted from LLM (gpt-4o-mini) via ‘action-by-action’ algorithm (Guan et al., 2023) on Logistics domain.