Planning in the Dark: LLM-Symbolic Planning Pipeline Without Experts

Sukai Huang[1], **Nir Lipovetzky**[1] **and Trevor Cohn**[1,2*]

[1]The University of Melbourne
[2] Google
sukaih@student.unimelb.edu.au, {nir.lipovetzky, trevor.cohn}@unimelb.edu.au

Abstract

Large Language Models (LLMs) have shown promise in solving natural language-described planning tasks, but their direct use often leads to inconsistent reasoning and hallucination. While hybrid LLM-symbolic planning pipelines have emerged as a more robust alternative, they typically require extensive expert intervention to refine and validate generated action schemas. It not only limits scalability but also introduces a potential for biased interpretation, as a single expert's interpretation of ambiguous natural language descriptions might not align with the user's actual intent. To address this, we propose a novel approach that constructs an action schema library to generate multiple candidates, accounting for the diverse possible interpretations of natural language descriptions. We further introduce a semantic validation and ranking module that automatically filter and rank the generated schemas and plans without expert-in-the-loop. The experiments showed our pipeline maintains superiority in planning over the direct LLM planning approach. These findings demonstrate the feasibility of a fully automated end-to-end LLM-symbolic planner that requires no expert intervention, opening up the possibility for a broader audience to engage with AI planning with less prerequisite of domain expertise.

Code — https://github.com/Sino-Huang/Official-LLM-Symbolic-Planning-without-Experts

Extended version — https://arxiv.org/abs/2409.15922

1 Introduction

The advent of Large Language Models (LLMs) has opened new avenues for solving natural language-described planning tasks (Kojima et al. 2022). However, direct plan generation using LLMs, while seemingly straightforward, has been criticized for inconsistent reasoning and hallucination, which undermines their reliability in critical planning scenarios (Valmeekam et al. 2022, 2023; Huang et al. 2024). In response, researchers have advocated for more robust approaches that combine the flexibility of LLMs with the correctness of symbolic planning to solve planning tasks (Pallagani et al. 2024; Oswald et al. 2024). To improve the soundness of generated plans, a hybrid LLM-symbolic planning pipeline has emerged. As shown in Figure 1, instead of relying solely on LLMs to generate sequences of action plans through in-context learning, this pipeline begins by leveraging LLMs to extract abstract symbolic action specifications from natural language descriptions, known as *action schemas*. These schemas define the essential components of an action in a structured format understandable by symbolic planners. Once these schemas are generated, a classical planner can take over to search for feasible plans that fulfill the task specifications (Liu et al. 2023; Silver et al. 2024; Guan et al. 2023; Kambhampati et al. 2024).

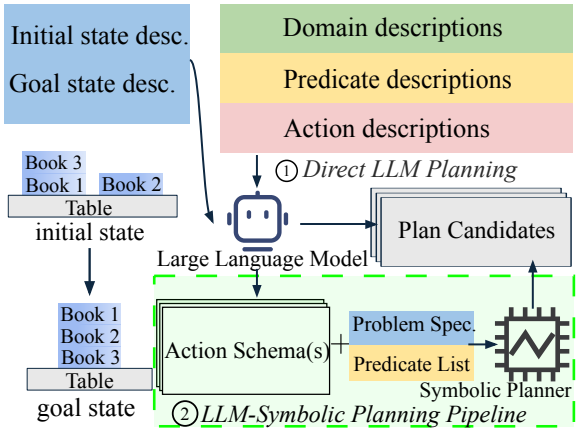

Figure 1: An overview of direct plan generation vs. LLM-symbolic planning pipelines.

Yet, this method is brittle, as a single missing or contradictory predicate in an action schema can prevent the planner from finding a valid plan. Thus, current pipelines often require multiple iterations of expert intervention to refine and validate the generated action schemas. For instance, Guan et al. (2023) reported that the expert took 59 iterations to fix schema errors for a single task domain. This process demands substantial time and expertise, which significantly hinders the ***scalability*** of the method. More critically, due to budget constraints, often only one expert is involved in the process. This creates a critical vulnerability: the potential for interpretation mismatch between the expert and the user. Experts, while knowledgeable, inevitably bring their own ***subjective interpretations*** to the task descriptions, often formal-

*Now at Google DeepMind

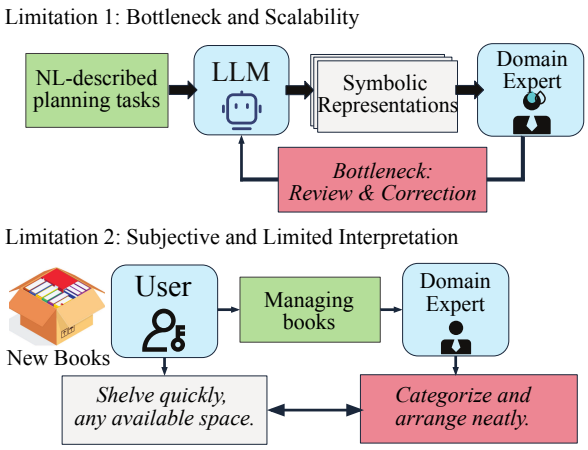

Figure 2: Illustration of the two limitations of expert-dependent LLM-symbolic planning pipelines

izing them in a single, specific way. This limits the system to a ***single perspective*** of the task. However, unlike formal language designed to have an exact, context-independent meaning, natural language inherently contains ambiguities that yield *diverse* valid interpretations of the same description. This ambiguity suggests that a straightforward, one-to-one mapping from natural to formal languages – a typical case when relying on a single expert – risks overlooking the interpretation that the user actually intended (Moravcsik 1983) (see Figure 2).

Regarding the issue with reliance on expert intervention, we propose a novel pipeline that eliminates this dependency. Specifically, our approach introduces two key innovations:

(1): We construct an *action schema library* to generate multiple candidates, a strategy that has been overlooked in prior work despite being a natural fit for capturing the inherent ambiguity in natural language. By leveraging this library, we also increase the likelihood of obtaining ***solvable*** action schema sets – those have at least one valid plan that can be found by a planner.

(2): We leverages sentence encoders[1] to automatically *validate and filter generated action schemas*. This module ensures that the generated schemas closely align with the task descriptions in the semantic space, effectively acting like expert feedback. Our experiments demonstrate that without expert intervention, our pipeline generates sound action plans competitive with direct LLM-based plan generation, even in short-horizon planning tasks. Importantly, our approach offers multiple schema sets and plan candidates, preserving the diversity of interpretations inherent in ambiguous natural language descriptions.

2 Related Work

Direct Plan Generation with LLMs: The use of LLMs for direct action plan generation has been explored across various domains, including embodied tasks (Wang et al. 2023; Xiang et al. 2024), and other language grounding environments (Ahn et al. 2022; Huang et al. 2022). These approaches are built upon the idea that LLMs' reasoning capabilities can be effectively elicited through in-context learning techniques, particularly the Chain-of-Thought (CoT) approach. CoT prompts the model to generate a series of intermediate reasoning steps before arriving at the final answer, resulting in more coherent and logically sound reasoning (Wei et al. 2022). Building upon CoT, Yao et al. (2024) proposed Tree-of-Thought (ToT) framework, which explores multiple reasoning pathways, generating diverse plans and ranking them based on self-verification heuristics. These heuristics are verbalized confidence scores produced by LLMs themselves, a method supported by studies showing that LLMs are effective as zero-shot ranking models (Lin et al. 2022; Hou et al. 2023; Zhuang et al. 2023).

Criticism and Hybrid Planning: Despite the promising results, researchers have raised concerns about the reliability and soundness of LLM-generated plans (Valmeekam et al. 2022, 2023; Huang et al. 2024). A critical issue highlighted by Kambhampati et al. (2024) is that planning and reasoning tasks are typically associated with System 2 competency, which involves slow, deliberate, and conscious thinking (Sloman 1996; Kahneman 2011). However, LLMs, being essentially text generators, exhibit constant response times regardless of the complexity of the question posed. This behavior suggests that no first-principle reasoning is occurring, contradicting the expectations for true planning capabilities. To this end, researchers have explored hybrid approaches. For instance, Thought of Search (Katz et al. 2024) involves the generation of successor function and goal test code by LLMs, followed by their execution within an external execution environment. The approach we focus on involves utilizing LLMs to generate symbolic representations of tasks, which are then processed by external symbolic planners to search for feasible plans (Liu et al. 2023; Guan et al. 2023). However, existing pipelines emphasize the necessity of expert intervention for action schema validation and refinement. While Kambhampati et al. (2024) proposed using LLMs as semi-expert critics to assess output quality, this approach still necessitates expert involvement for final decision-making. In contrast, our work strives to reduce the dependency on expert intervention, offering a more accessible approach to hybrid LLM-symbolic planning that also addresses the inherent ambiguity in natural language descriptions.

3 Problem Setting and Background

We consider a scenario where an agent generates action plans for natural language-described planning tasks. A task description typically consists of: (1) a domain description outlining general task information and possible high-level actions, and (2) a problem instance description specifying the initial and goal states. The study of LLM-symbolic planning pipelines is grounded in the formal framework of classical planning, which relies on symbolic representations of planning tasks. These representations are typically expressed using the Planning Domain Definition Language (PDDL) (Aeronautiques et al. 1998; Haslum et al. 2019).

[1]Sentence encoders are neural network models that transform sentences into vector representations, capturing semantic meaning

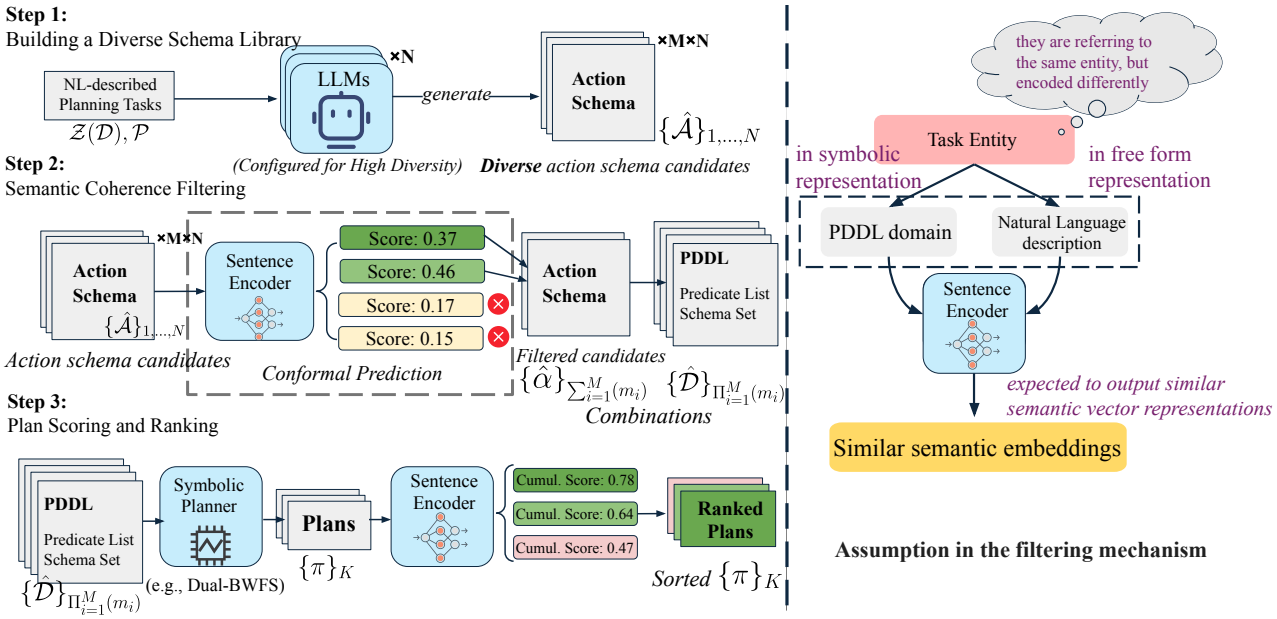

Figure 3: An overview of the proposed pipeline, it first constructs diverse action schema candidates to cover various interpretations of the natural language descriptions. Then, it filters out low-confidence candidates to ensure the generation candidates are semantically aligned with the descriptions. Lastly, it produces and ranks multiple plans using a symbolic planner. The filtering mechanism is grounded in the concept of semantic equivalence across different representations of the same content.

In brief, a PDDL description is defined by $\langle \mathcal{D}, \Pi_{\mathcal{D}} \rangle$, where:

- $\mathcal{D} = \langle \mathcal{P}, \mathcal{A} \rangle$ is the domain specification: $\mathcal{P}$ is the set of predicates that can either hold true or false, and $\mathcal{A}$ is the set of action schemas. Each action schema $\alpha \in \mathcal{A}$ is defined as a tuple $\alpha = \langle par, pre, eff \rangle$, where par details the parameters, and pre and eff are the preconditions and effects, respectively. Both pre and eff are typically expressed as conjunctive logical expressions using predicate logic.
- $\Pi_{\mathcal{D}} = \langle \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is the problem instance: $\mathcal{O}$ is the set of objects to interact with, $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal state that the agent needs to achieve.

A solution to the planning task is a sequence of grounded actions ($\pi = (a_0, ..., a_n)$) that transforms the initial state $\mathcal{I}$ to the goal state $\mathcal{G}$. Each grounded action a_i is an instantiation of an action schema $\alpha \in \mathcal{A}$ and predicates, where the parameters in α are replaced with specific objects from $\mathcal{O}$.

To bridge natural language descriptions and formal planning representations, we introduce a natural language proxy layer, denoted as $\mathcal{Z}(\cdot)$, for these task specifications. For example, $\mathcal{Z}(\mathcal{D})$ represents the natural language equivalent of the domain specification $\mathcal{D}$. The two approaches, *direct LLM planning* and *LLM-symbolic planning*, can then be expressed in Eq 1 and Eq 2, respectively:

$$\pi \sim P_{\text{LLM}}(\cdot \mid \mathcal{Z}(\mathcal{D}), \mathcal{Z}(\Pi_{\mathcal{D}})) \tag{1}$$

$$\hat{\mathcal{A}} \sim P_{\text{LLM}}\left(\cdot \mid \mathcal{Z}(\mathcal{D})\right); \Pi_{\mathcal{D}} \sim P_{\text{LLM}}\left(\cdot \mid \mathcal{Z}(\Pi_{\mathcal{D}})\right)$$

$$\pi = f\left(\langle \mathcal{P}, \hat{\mathcal{A}} \rangle, \Pi_{\mathcal{D}}\right) \tag{2}$$

In these equations, $P_{\text{LLM}}(\cdot)$ represents the generation process of LLMs, and f is the symbolic planner that search for sound plans. While we largely adhere to the problem setting of previous research (e.g., Liu et al. (2023), Guan et al. (2023)), we introduce a crucial refinement by specifying a precise predicate set ($\mathcal{P}$) for each domain descriptions. This controlled setting addresses a key challenge in evaluating across different methodologies. Without a standardized predicate set, variations in domain understanding can lead to diverse and potentially incomparable outputs, hindering meaningful evaluation.

4 Methodology

As illustrated in Figure 3, the proposed pipeline stands in contrast to existing expert-dependent approaches and consists of three key steps: *(1) Building a Diverse Schema Library (§4.1), (2) Semantic Coherence Filtering (§ 4.2) and (3) Plan Scoring and Ranking (§ 4.4).*

4.1 Building a Diverse Schema Library

A key challenge in translating natural language descriptions into symbolic action schemas is the inherent ambiguity of language itself. Different interpretations of the same description can lead to variations in action schemas, impacting the downstream plan generation process. To ensure we explore a wide range of interpretations and effectively cover the user's intent, we utilize multiple LLM instances, denoted as $\{P^1_{\text{LLM}}, P^2_{\text{LLM}}, ..., P^N_{\text{LLM}}\}$, and set their temperature hyperparameter high to encourage diverse outputs. Each will then generate its own set of action schemas $\hat{\mathcal{A}}_i \sim P^i_{\text{LLM}}(\cdot \mid$

$\mathcal{Z}(\mathcal{D})$), where $\hat{\mathcal{A}}_i = (\hat{\alpha}_{i1}, \hat{\alpha}_{i2}, ..., \hat{\alpha}_{iM})$. Here, $\hat{\alpha}_{ij}$, where $i \in [1, ..N]$ and $j \in [1, ..., M]$, represents the generated action schema of j-th action in the domain by the i-th LLM instance.

The generated schemas $\hat{\alpha}_{ij}$ from all models are then aggregated into a single library. Since each domain comprises M actions, a "set" of action schemas refers to a complete collection where each action in the domain is associated with one corresponding schema. Therefore, all possible combination of action schemas within the library can generate approximately $\binom{N}{1}^M$ different sets of action schemas.

In addition, existing pipelines rely heavily on expert intervention, partly because individual LLMs struggle to generate *solvable* sets of schemas – those that a planner can successfully use to construct a plan. This reliance becomes even more pronounced as the number of actions increases, with the probability of obtaining a solvable set of schemas from a single LLM diminishing exponentially. In contrast, our approach, by constructing a diverse pool of action schema sets, substantially improves the probability of finding a solvable set. Our analysis (detailed in Appendix A) demonstrates that, under reasonable assumptions, this probability can increase from less than 0.0001% with a single LLM to over 95% when using multiple LLM instances.

Note that the *solvability* of a set of action schemas can be efficiently verified by leveraging the *completeness* feature of modern symbolic planners. If a plan can be found for a given problem using the generated schemas, the set is deemed solvable. Importantly, modern symbolic planners have advanced capabilities that allow them to efficiently reject unsolvable schema sets. This is achieved by the ability to prove delete-free reachability in polynomial time (Bonet and Geffner 2001). Furthermore, modern planners are designed to operate efficiently on multithread CPU and the efficiency of the process should not be a cause for concern. See Appendix D for more details.

4.2 Semantic Coherence Filtering

The previous method alone faces two limitations. First, as task complexity grows, the "brute-force" approach of combining and evaluating all possible sets becomes increasingly inefficient. Second, solvability does not guarantee semantic correctness – schemas may not accurately reflect the task descriptions, potentially leading to incorrect or nonsensical plans. Therefore, it is crucial to implement a filtering mechanism that autonomously assesses the semantic correctness of individual action schemas, filtering out low-quality candidates before they enter the combination process.

Our approach is grounded in the concept of semantic equivalence across different representations of the same content, as discussed by Weaver (1952) in his memorandum "Translation." Weaver emphasized that the most effective way to translate between languages is to go deeper to uncover a shared "common base of meaning" between language representations, illustrating this by noting that "a Russian text is really written in English, but it has been encoded using different symbols." This principle is crucial in our context, where task descriptions in natural language and their corresponding structured symbolic representations should exhibit high semantic similarity, reflecting the same shared meaning despite different syntactic forms (see right side of Figure 3).

Recent developments in language models as code assistants (Chen, Tworek et al. 2021; Rozière, Gehring et al. 2024) further support this assumption, demonstrating that these models can decode the underlying semantics of structured symbolic representations. Inspired by this, we propose a filtering step that leverages a sentence encoder $E(\cdot)$ to generate embeddings for both the action descriptions $E(\mathcal{Z}(\alpha))$ and the generated schemas $E(\hat{\alpha})$. Then, we compute the cosine similarity between these embeddings to quantify semantic relatedness and filter out action schemas with low scores.

Specifically, we employ a conformal prediction (CP) framework (see Appendix B) to statistically guarantee that true positive action schema candidates have a high probability of being preserved while minimizing the size of the filtered set (Sadinle et al. 2019). In this process, a threshold $\hat{q}$ will be calculated based on a user-specified confidence level $1 - \epsilon$. Action schemas with cosine similarity scores below this threshold are filtered out from the library.

This process (illustrated in *step 2* of Figure 3) significantly reduces the *number of candidate sets of action schemas* to $\Pi_{i=1}^{M}(m_i)$, where m_i is the number of action schemas that pass the semantic validation for the i-th action. This prefiltering approach not only reduces the computational load on the symbolic planner, increasing efficiency, but also ensures that generated schemas closely align with the semantic meaning of the task descriptions.

4.3 Finetuning with Manipulated Action Schemas

Hard negative samples have been shown to enhance representation learning by capturing nuanced semantic distinctions (Robinson et al. 2023). In our context, we found that structured action schemas are particularly ideal for generating hard negatives. By manipulating predicates in the precondition or effect expressions of true action schemas, we create hard negatives with subtle differences. During finetuning, a triplet loss function is employed, where each training sample consists of a triplet: the natural language description of an action $(\mathcal{Z}(\alpha))$, the true action schema (α), and a negative sample (α^{neg}) (see Figure 4). A negative sample is of three types – *(1) Easy Negatives*: action schemas from other planning domains (inter-domain mismatch); *(2) Semi-Hard Negatives*: action schemas from the same domain but referring to different actions (intra-domain mismatch); and *(3) Hard Negatives*: As shown in Table 1, we employ four types of manipulations – swap, negation, removal, and addition – to manipulate the reference[2] action schema of the domain.

[2]A reference domain model is only used for reference when we create manipulated versions of action schemas. We do this to recognize that natural language can be interpreted in various ways, rather than presupposing a one-to-one correspondence with a single ground truth schema, as discussed in Sec 1.

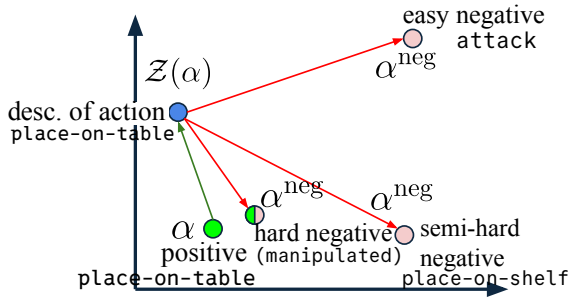

Figure 4: Finetuning sentence encoder with triplet loss.

Manipulation Type	Description
Swap	Exchanges a predicate between preconditions and effects
Negation	Negates a predicate in either preconditions or effects
Removal	Removes a predicate from either preconditions or effects
Addition	Adds mutually exclusive (mutex) predicates to preconditions or effects (Helmert 2009)

Table 1: Types of Manipulations for Generating Synthesized Hard Negative Action Schemas in Training Data. Mutexes are predicates that cannot be true simultaneously, e.g., one cannot hold a book and have it on a table simultaneously.

Through this process, the sentence encoder learns to embed natural language descriptions closer to their corresponding action schemas while distancing them from negative samples in the semantic space.

4.4 Plan Generation and Ranking

Action schemas that more accurately represent the intended tasks described in natural language are likely to yield higher-quality, more reliable plans. Leveraging this causal relationship, we assess and rank the generated plans based on the cumulative semantic similarity scores of their constituent action schemas. Specifically, we feed each solvable set of action schemas into a classical planner, which generates a corresponding plan. Then, the ranking score for a plan is calculated as $\sum_{i=1}^{M} \frac{E(\mathcal{Z}(\alpha_i)) \cdot E(\hat{\alpha_i})}{\|E(\mathcal{Z}(\alpha_i))\| \|E(\hat{\alpha_i})\|}$, where $\mathcal{Z}(\alpha_i)$ is the natural language description of the i-th action in the domain, $\hat{\alpha_i}$ is the corresponding generated action schema and $E(\cdot)$ is already defined in Sec 4.2. It ensures that the structured symbolic model comprising the plans are semantically aligned with the descriptions of the planning domain (see *step 3* in Figure 3). Furthermore, this approach allows for optional **lightweight expert intervention** as a final, non-iterative step. By presenting the ranked schema sets and their corresponding plans, experts can determine the most appropriate one, providing a balance between autonomy and expert guidance.

Overall, our pipeline bridges the gap between ambiguous task descriptions and the precise requirements of symbolic planners. By generating a diverse pool of action schemas and leveraging semantic similarity for validation and ranking, we achieve two key advancements. First, we reduce the dependency on expert intervention, making the process more accessible and efficient. Second, we preserve the inherent ambiguity of natural language, offering users multiple valid interpretations of the task and their corresponding plans.

5 Experiments

Our experiments test the following hypotheses: **(H1)** Semantic equivalence across different representations, as discussed by Weaver, holds true in our context. **(H2)** Ambiguity in natural language descriptions leads to multiple interpretations. **(H3)** Our pipeline produces multiple solvable candidate sets of action schemas and plans without expert intervention, providing users with a range of options. **(H4)** Our pipeline outperforms direct LLM planning approaches in plan quality, demonstrating the advantage of integrating LLM with symbolic planning method. See Appendix for other experiments outside the scope of these hypotheses.

5.1 Experimental Setup

Task and Model Setup. We introduces several key enhancements that distinguish it from previous work. *(1) Novel Test Domains:* We carefully selected three test domains ensuring they are unfamiliar to LLMs – **Libraryworld**: a modified version of the classic Blockworld domain; **Minecraft**: resource gathering and crafting domain inspired by the game Minecraft; and **Dungeon**: a domain originally proposed by Chrpa et al. (2017). This approach addresses a significant issue: many IPC[3] domains have likely been leaked into LLM training data (see Appendix C). For training and calibration of the sentence encoder, we used domains from IPC and PDDLGym (Silver and Chitnis 2020). *(2) LLM Selection:* We use the open-source GLM (Hou et al. 2024) over proprietary models like GPT-4, aligning with our commitment to accessible planning systems. *(3) Ambiguity Examination:* We tested our pipeline on two types of task descriptions to assess the impact of ambiguity – (a) ***detailed*** descriptions following the established style of Guan et al. (2023), and (b) ***layman*** descriptions provided by five non-expert participants[4] who, unfamiliar with PDDL, described the domains and actions based on reference PDDL snippets. *(4) Symbolic Planner:* We used *DUAL-BWFS* (Lipovetzky and Geffner 2017) planner for plan generation as well as checking if the generated schema sets are solvable. *(5) LLM Prompt Engineering:* We use the CO-STAR and CoT framework to guide LLMs in generating outputs (see Appendix E).

Baselines. We evaluate our pipeline against two key baselines: *(1) The previous LLM-symbolic planning pipeline* proposed by Guan et al. (2023), which involves expert intervention for action schema validation and refinement; and *(2) Direct LLM-based planning* using Tree-of-Thought (ToT) (Yao

[3]International Planning Competition, a benchmark event for automated planning systems using PDDL.

[4]Business school students with no prior knowledge of PDDL programming or computational logic

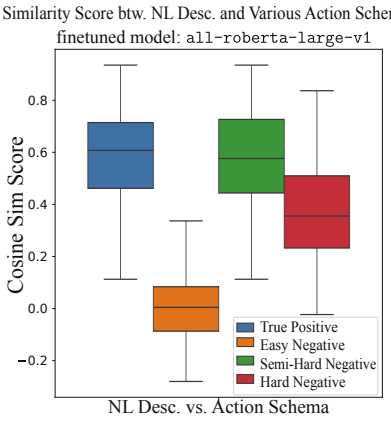

Figure 5: The sentence encoder enhances the identification of mismatched pairs by fine-tuning with negative samples.

Model	Mech.	Expert Input #	Heuristic Type	Soundness w.r.t. Schemas
Tree-of-Thought (Yao et al. 2024)	LLM	0	Self Verification	No
Guan et al. (2023)	Hybrid	≈ 59	Expert Validation	Yes
Ours	Hybrid	≤ 1	Semantic Sim. Scores	Yes

Table 2: Contrasts Our Pipeline with Existing Works. Note that the property of generating sound (logical correct) plans has been highlighted as a feature of the hybrid planner in prior work (Liu et al. 2023; Guan et al. 2023). However, there is no guarantee that the schemas are fully correct w.r.t. what the user actually wants. Thus, we are weakening the property to soundness w.r.t. schemas.

et al. 2024), which generates multiple plans and ranks them based on self-verification heuristics.

5.2 Semantic Equivalence Analysis

To investigate **H1**, we initially assessed the cosine similarity of sentence embeddings for pairs of action schemas and their corresponding natural language descriptions, both when they were matched and when they were mismatched. We employed two pre-trained, extensive sentence encoders: *text-embedding-3-large* and *sentence-t5-xl*. These models, without any fine-tuning, demonstrated higher cosine similarity for matched pairs compared to mismatched ones. This finding suggests that the ability to detect such equivalence is an inherent feature of high-quality sentence embedding models, not merely an *artifact* of fine-tuning. However, OpenAI *text-embedding-3-large* model is bad for its accessibility, a lightweight encoder *all-roberta-large-v1* allows for better speed and improved accuracy through fine-tuning, which is good in practice. The performance of the fine-tuned *roberta* model is shown in Figure 5. The substantial improvement in the model's capacity to identify hard negatives – mismatched pairs with subtle differences – is a direct result of our dedicated training weights allocation. We deliberately designed our training data selection to include a ratio of easy, semi-hard, and hard negatives as $[0.0, 0.4, 0.6]$, respectively (see Appendix E.7). This ratio was strategically chosen to concentrate on hard negatives, as LLMs are more likely to make hard-negative mistakes when generating action schemas. By prioritizing hard negatives in our training dataset, we aimed to enhance the model's ability to filter out low-quality action schemas during the semantic coherence filtering step.

5.3 Pipeline Performance and Efficiency

Our pipeline's performance and efficiency are highlighted through several key observations. Firstly, the use of action schema library effectively produces *solvable* action schema sets without requiring expert-in-the-loop. Notably, deploying 10 LLM instances is sufficient to generate solvable schema sets for all test domains, supporting **H3**. Secondly, Figure 6 reveals a clear pattern: when confronted with inherently ambiguous ***layman*** descriptions from non-expert participants, our pipeline generates a significantly increased number of distinct solvable schema sets (e.g., from 3419 to 8039 when LLM# = 10 w/o CP), thereby supporting **H2**. This increase is primarily attributed to the diverse selection of predicates within the action schemas. Each predicate selection reflects a different interpretation of the problem, with each schema set *emphasizing distinct features* deemed critical for planning.

For instance, in the *Libraryworld* domain, we observed that some schema sets generated by some LLM instances take into account the 'category' property of books when constructing actions such as stacking books on a shelf. This means that, according to these schema sets, only books within the same category can be stacked together, which is a more organized way of arranging books. Consequently, this leads to different planning outcomes that reflect the varied interpretations of the user query at hand, which are a direct result of the ambiguity present in the layman's description and the flexibility it provides to LLMs in making such choices.

The pipeline's ability to generate a range of potential interpretations in response to ambiguous inputs is a critical advantage. It ensures that all intended aspects of the user's description can be captured, even when the description is imprecise or incomplete.

Thirdly, the integration of conformal prediction in the filtering step demonstrates a significant improvement in efficiency, as evidenced by Figure 6. With the confidence level $1 - \epsilon$ set to 0.8, the pipeline filtered out a large number of candidates, reducing the total number of combinations to 3.3% of the original (1051 out of 31483) but meanwhile, the ratio of solvable schemas (verified by the planner) increased from 10.9% to 23.0%. This result strongly supports **H3**, highlighting the pipeline's ability to efficiently generate solvable and semantically coherent schema sets. See Table 2 for a comprehensive comparison of our pipeline with existing LLM-based planning approaches. Notably, the initial low ratio of solvable schema sets (10.9%) under-

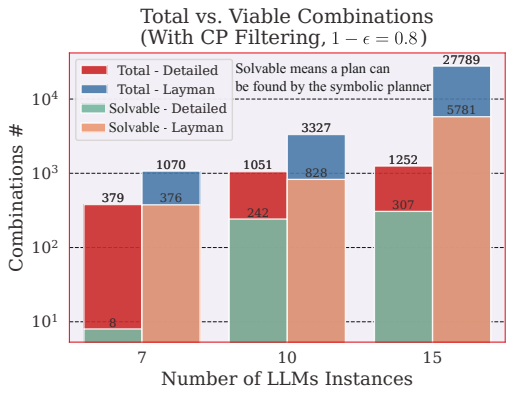

Figure 6: With CP, a large number of candidates are pruned, thereby improving efficiency.

	Rank 1st	Rank 2nd	Rank 3rd	Rank 4th	Rank 5th	Avg.
Gold	14	4	4	1	1	1.79
Ours	4	18	11	5	10	2.97
ToT	6	2	9	18	13	3.62

Table 3: Blind plan ranking eval.: Four assessors compared the top two plans from each approach to gold plans.

scores the challenge faced within the LLM-symbolic planning paradigm, which may explain why expert intervention has been a common practice in the past.

5.4 Human Evaluation on Plan Quality

To further validate our approach, we conducted a human evaluation comparing the top two plan candidates generated by our pipeline against those from the ToT framework and a gold-standard plan derived from the reference PDDL domain model. Four expert assessors with extensive PDDL experience ranked the plans based on their feasibility in solving the given problems. The results, summarized in Table 3, clearly support **H4**.

For a deeper insight into our pipeline's capabilities, we specifically tested the Sussman Anomaly, a well-known planning problem that requires simultaneous consideration of multiple subgoals, as solving them in the wrong order can undo previous progress (see Figure 1). Our results showed that ToT-style approaches using various LLMs, including state-of-the-art models like GPT-4o, consistently fail to solve this problem. The failure arises from the mistaken assumption that the first subgoal mentioned (i.e., placing book 1 on top of book 2) should be addressed first, leading to incorrect plans. Interestingly, GPT-3.5 and GPT-4o exhibited different behaviors when faced with this problem. While GPT-3.5 consistently, yet incorrectly, asserted it had completed the problem, GPT-4o occasionally exhibited awareness of the plan's incompleteness. However, even with this heightened awareness, GPT-4o was unable to identify the correct path within the given depth limit. Notably, ToT-style approaches reveals a critical limitation where high verbalized confidence scores does not necessarily translate to plan validity. In contrast, our pipeline generates a range of plans, including suboptimal ones, but excels at identifying and prioritizing the most promising candidates through its ranking process that is based on the cumulative cosine similarity scores of generated action schemas. By strictly adhering to semantic alignment between these schemas and natural language descriptions, and by using a symbolic planner, the system avoids being misled by the tendency – observed in both humans and LLMs – to reason in a linear manner. This tendency involves prioritizing subgoals based on their order of appearance rather than considering their underlying logical dependencies. Such linear reasoning can lead to noninterleaved planning, where subgoals are tackled in the order they are presented and each must be fully completed before addressing the next one, which is a pitfall in complex planning problems like the Sussman Anomaly.

5.5 Failure Case Analysis

Schema Set with No Plan Found: We encountered instances where no solvable action schema set was generated, primarily due to limitations in the LLM's reasoning capabilities. The use of open-source LLMs, while more accessible, may result in a lower success rate compared to more advanced proprietary models like GPT-4o. Specifically, with 7 LLM instances, we observed occasional failures of generating solvable sets action schemas for the *libraryworld* and *minecraft* domains. Nevertheless, solvable schema sets were consistently obtained across all domains when the number of LLM instances was increased to 10 (see Appendix F for a breakdown of schema set yield by LLM instance count).

Unexpected Preference: In the *Dungeon* domain, human assessors unexpectedly preferred ToT-generated plans over both the reference plan and the proposed pipeline's plans. Further analysis revealed that the ToT plans consistently included a step: *grabbing a sword*. Interestingly, grabbing a sword was not a necessary step for solving the given problem. Consequently, symbolic planners, focused on optimal pathfinding, excluded this step from their plans. However, this "unnecessary" step of acquiring a sword aligns with common strategies in Dungeon games, where players typically prioritize preparedness. Thus, this action strongly appealed to human assessors, causing them to rank the ToT-generated plans higher.

6 Conclusion

Our work presents a 3-step pipeline that learn symbolic PDDL models over ambiguous natural language descriptions and generated multiple ranked plan candidates. Our findings demonstrate that a full end to end hybrid planner is possible without expert intervention, paving the way for democratizing planning systems for a broader audience. One limitation in this work is the lack of direct evaluation methods for assessing the quality of generated action schema sets. Metrics like "bisimulation" (Coulter et al. 2022) or "heuristic domain equivalence" (Oswald et al. 2024) require the generated schema sets to have the same action parameters as a predefined reference set. This approach doesn't suit our context, where action parameters are flexible and inferred in real-time from natural language descriptions.

Acknowledgements

Sukai Huang is supported by Melbourne Research Scholarship established by The University of Melbourne.

This research was supported by The University of Melbourne's Research Computing Services and the Petascale Campus Initiative.

References

Aeronautiques, C.; Howe, A.; Knoblock, C.; McDermott, I. D.; Ram, A.; Veloso, M.; Weld, D.; Sri, D. W.; Barrett, A.; Christianson, D.; et al. 1998. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Ho, D.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jang, E.; Ruano, R. M. J.; Jeffrey, K.; Jesmonth, S.; Joshi, N. J.; Julian, R. C.; Kalashnikov, D.; Kuang, Y.; Lee, K.-H.; Levine, S.; Lu, Y.; Luu, L.; Parada, C.; Pastor, P.; Quiambao, J.; Rao, K.; Rettinghouse, J.; Reyes, D. M.; Sermanet, P.; Sievers, N.; Tan, C.; Toshev, A.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; and Yan, M. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Conference on Robot Learning*.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2): 5–33.

Chen, M.; Tworek, J.; et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Chrpa, L.; et al. 2017. The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. *AI Magazine*, 38(1): 104–106.

Coulter, A.; Ilie, T.; Tibando, R.; and Muise, C. 2022. Theory Alignment via a Classical Encoding of Regular Bisimulation. In *Proceedings of the Workshop on Knowledge Engineering for Planning and Scheduling (KEPS) at ICAPS*. ICAPS.

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. *Advances in Neural Information Processing Systems*, 36: 79081–79094.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; Muise, C.; Brachman, R.; Rossi, F.; and Stone, P. 2019. *An introduction to the planning domain definition language*, volume 13. Springer.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5-6): 503–535.

Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; and Zhao, W. X. 2023. Large Language Models are Zero-Shot Rankers for Recommender Systems. In *European Conference on Information Retrieval*.

Hou, Z.; Niu, Y.; Du, Z.; Zhang, X.; Liu, X.; Zeng, A.; Zheng, Q.; Huang, M.; Wang, H.; Tang, J.; and Dong, Y. 2024. ChatGLM-RLHF: Practices of Aligning Large Language Models with Human Feedback. arXiv:2404.00934.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; and Liu, T. 2024. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Kahneman, D. 2011. *Thinking, fast and slow*. macmillan.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Forty-first International Conference on Machine Learning*.

Katz, M.; et al. 2024. Thought of Search: Planning with Language Models Through The Lens of Efficiency. arXiv:2404.11833.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Lin, S. C.; et al. 2022. Teaching Models to Express Their Uncertainty in Words. *Trans. Mach. Learn. Res.*, 2022.

Lipovetzky, N.; and Geffner, H. 2017. Best-First Width Search: Exploration and Exploitation in Classical Planning. In *AAAI Conference on Artificial Intelligence*.

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

Moravcsik, J. M. 1983. Natural languages and formal languages: a tenable dualism. In *Language, Logic and Method*, 225–239. Springer.

Oswald, J.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024. Large Language Models as Planning Domain Generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 423–431.

Pallagani, V.; Roy, K.; Muppasani, B.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In *International Conference on Automated Planning and Scheduling*.

Robinson, J. D.; Chuang, C.-Y.; Sra, S.; and Jegelka, S. 2023. Contrastive Learning with Hard Negative Samples. In *International Conference on Learning Representations*.

Rozière, B.; Gehring, J.; et al. 2024. Code Llama: Open Foundation Models for Code. arXiv:2308.12950.

Sadinle, M.; et al. 2019. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525): 223–234.

Silver, T.; and Chitnis, R. 2020. PDDLGym: Gym Environments from PDDL Problems. In *ICAPS Workshop on*

Bridging the Gap Between AI Planning and Reinforcement Learning (PRL).

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L.; and Katz, M. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20256–20264.

Sloman, S. A. 1996. The empirical case for two systems of reasoning. *Psychological Bulletin*, 119: 3–22.

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

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In *Neural Information Processing Systems*.

Wang, Z.; Cai, S.; Chen, G.; Liu, A.; Ma, X.; and Liang, Y. 2023. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. In *Neural Information Processing Systems*.

Weaver, W. 1952. Translation. In *Proceedings of the Conference on Mechanical Translation*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xiang, J.; Tao, T.; Gu, Y.; Shu, T.; Wang, Z.; Yang, Z.; and Hu, Z. 2024. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zhuang, S.; Liu, B.; Koopman, B.; and Zuccon, G. 2023. Open-source Large Language Models are Strong Zero-shot Query Likelihood Models for Document Ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 8807–8817.