ALPSBench: Can Large Language Models Reason Their Way Through Planning Formalisms?

Marcus Tantakoun[1,2], **Gabrielle Garey**[2], **Christian Muise**[1,2], **Xiaodan Zhu**[1,3]
[1]Ingenuity Labs Research Institute, Queen's University
[2]School of Computing, Queen's University
[3]Department of Electrical and Computer Engineering, Queen's University
{20mt1, christian.muise, xiaodan.zhu}@queensu.ca

Abstract

The rapid emergence of increasingly powerful Large Language Models (LLMs), especially those augmented with "thinking"-based architectures, has fueled enthusiasm toward specification-driven planning paired with external solvers. However, it remains unclear if these models can reliably encode world dynamics with the semantic accuracy needed for successful plan execution in novel domains. To this end, we introduce **ALPSBench**, a benchmark for evaluating LLMs' reasoning capabilities in planning model construction. ALPSBench consists of 24 novel PDDL domains that serve as the basis for 2 complementary evaluations: *ALPSGEN* and *ALPSQA*, which target PDDL domain generation and diagnostic repair. Our extensive evaluation of 9 state-of-the-art LLMs from 3 different families highlights the significant gap in the reasoning capabilities over such formalisms. Our results show that while modern LLMs achieve high syntactic validity and executable plans in unseen PDDL domains, the plans are rarely validated against their ground-truth counterparts. Additionally, across all models, when given less specific diagnostic prompts for detecting faulty PDDL actions, their performance degrades significantly.

	Domain	# Pred.	# Actions	Est. Tokens
Easy	Campfire	9	8	1062
	RoboDelivery	7	7	885
	GardenMan	7	7	851
	Laundromat	8	6	845
	Librarian	9	7	451
	Recycling	10	7	802
	HazardShips	9	4	667
	TrafficJam	7	4	549
Medium	CoffeeShop	12	6	946
	Chipline	9	8	1087
	Diceworld	11	8	1273
	Firefighter	10	7	1157
	Grizzwork	10	8	1235
	PizzaMan	13	10	1265
	ColorPush	7	3	622
	Robofab	10	5	710
Hard	Bucketworld	7	5	856
	Colormix	10	8	1659
	Contamination	11	6	880
	KitchenSpread	8	3	590
	Luxgrid	4	4	823
	PostApoc	13	5	1160
	Verticalfarm	10	5	1048
	Watersupply	4	3	573

Table 1: Overview of constructed PDDL domains in ALPSBench. For each domain, we report the number of lifted predicates, actions, and the estimated number of tokens.

1 Introduction

Recent advances in the planning community demonstrate the growing capability of Large Language Models (LLMs) to produce syntactically well-formed PDDL specifications, often without the need for fine-tuning or carefully engineered prompting (Yu et al. 2025). While this potential has brought light towards automating planning-model construction into real-world applications, it also sharpened focus on more critical challenges. In particular, it remains an open question as to whether these models can reliably encode the intended world dynamics that support successful and correct plan execution when applied to **previously unseen domains**.

There remains significant debate over whether LLMs are genuinely *reasoning* about planning dynamics when constructing PDDL domains, or merely producing syntactically plausible specifications with limited semantic grounding. A common evaluation strategy compares LLM-generated domains to their ground-truth counterparts. However, two critical issues undermine the reliability of this approach. First, due to the scarcity of publicly available PDDL datasets, the International Planning Competition (IPC) benchmarks have been exhaustively reused across the community (Liu et al. 2023; Gestrin, Kuhlmann, and Seipp 2024). Their widespread circulation increases the likelihood that these domains appear in LLM training corpora, leading to memorization effects and inflated performance estimates (Katz et al. 2025). [1] Second, the reliability of existing annotated datasets is increasingly uncertain. Hu et al. (2025) proposed web-scraping to compile a large suite of PDDL domains. However, many were extracted from GitHub repositories and lacked paired problem files, raising concerns about their quality and correct plan executability. In contrast, Stein and Koller (2023) introduced an LLM-based pipeline to translate PDDL domains to natural language (NL) descriptions. While this does provide scalability, these translations are often error-prone, frequently misinterpreting variable names or domain semantics. In many cases, neither the examples nor their corresponding ground truth is rigorously verified. This underscores the need for novel, high-quality domains that are carefully curated and verified by human annotators.

[1]Chatbots (e.g., ChatGPT) now often reproduce entire domains (e.g., BLOCKSWORLD or GRIPPERS) without explicit prompting.

To address these gaps, we introduce a new PDDL domain and annotation benchmark, **ALPSBench** (Automating Language to Planning Specifications). ALPSBench (Table 1) consists of 24 hand-crafted, novel PDDL domains that supports two complementary datasets: 1) **ALPSGEN**, which pairs each domain with natural language (NL) descriptions of varying contextual granularity; and 2) **ALPSQA**, a Q&A-style assessment consisting of 360 boolean (BOOL), multiple-choice (MCQ), and generative (GEN) task-based questions in 8 of the 24 domains. In this paper, we are interested in evaluating the *generative* capabilities of LLMs on unseen domains and conducting fine-grained analysis of their ability to *diagnose* faulty action schemas.

Our contributions can be summarized as follows:

- We introduce **ALPSBench**, a benchmark that consists of 24 hand-crafted novel PDDL domains that supports two datasets: ALPSGEN and ALPSQA to evaluate the reasoning capabilities over PDDL generation and repair.
- To support our benchmark, we evaluate 9 frontier LLMs across 3 different model families with zero and few-shot prompting configurations.
- To advance LLM-based domain generation, we investigate several modelling strategies, including our proposed **ALPS-ITER** framework.

2 Background

2.1 The Planning Domain Definition Language

In this paper, we focus on the Planning Domain Definition Language (PDDL) (McDermott et al. 1998). PDDL is a widely adopted formalism for expressing planning domains and problems in Automated Planning. It extends the classical STRIPS representation by allowing lifted actions, typed objects, and predicates, that enable more general and reusable action definitions. Its representation consists of the *Domain* file ($\mathbb{DF}$) and a complementary *Problem* file ($\mathbb{PF}$).

The **Domain File** ($\mathbb{DF}$) specifies the universal dynamics of the planning environment, independent of any specific instance. Formally, a domain can be defined as a 4-tuple: $\mathbb{DF} = \langle \mathcal{T}, \mathcal{C}, \mathcal{P}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the hierarchy of types that classifies objects $\mathcal{O}$; $\mathcal{C}$ is the set of typed constants that are globally defined, fixed objects that exist in every problem within the domain; $\mathcal{P}$ is the set of lifted predicates, each defining a relation over typed arguments and implicitly defining the state space; and $\mathcal{A}$ is the set of lifted actions available to the planner. Each action schema $a \in \mathcal{A}$ is defined as: $a = \langle \text{par}(a), \text{pre}(a), \text{eff}^+(a), \text{eff}^-(a) \rangle$. Here, par(a) is the set of typed parameters; pre(a) is the set of preconditions that must hold for the action to be applicable; eff^+(a) and eff^-(a) are the add and delete effects, respectively. Grounded predicates are those with variables replaced with objects $o \in \mathcal{O}$, that form the set: $\mathcal{P}_g = \{p(o_1, ..., o_k) \mid p \in \mathcal{P}, o_i \in \mathcal{O}\}$. The state transition function $\delta : S \times \mathcal{A}_g \to S$ computes the successor state, where $\mathcal{A}_g$ denotes the set of grounded actions obtained by instantiating each lifted action schema with concrete objects. The transition for a specific grounded action a_g is given by:

$$\delta(s, a_g) = (s \setminus \text{eff}_g^-(a_g) \cup \text{eff}_g^+(a_g))$$

The **Problem File** ($\mathbb{PF}$) grounds the domain with concrete objects, an initial state, and goals for a specific planning instance. A problem is defined as: $\mathbb{PF}_i = \langle \mathcal{O}_i, s_0^i, G_i \rangle$, where $\mathcal{O}_i$ is the set of typed objects used in instance i; $s_0^i \subseteq \mathcal{P}_g(\mathcal{O}_i)$ is the initial state—the set of grounded predicates that are true; and $G_i \subseteq \mathcal{P}_g(\mathcal{O}_i)$ specifies the desired goal conditions.

A grounded action a_g is *applicable* in some state s when all of its grounded preconditions hold, i.e., when $\text{pre}_g(a_g) \subseteq s$. A valid plan $\pi_i = \langle a_1, a_2, ..., a_n \rangle$ for $\mathbb{PF}_i$ with $\mathbb{DF}$ is a sequence of such actions that incrementally transforms the initial state s_0^i into one satisfying G_i. The precise mechanics of action application are trivial for the remainder of the paper. Essentially, PDDL defines a structured symbolic interface for external planners, denoted Θ, that can take these two files as input and generate a plan: $\pi_i = \Theta(\mathbb{DF}, \mathbb{PF}_i)$.

2.2 PDDL and Large Language Models

Beyond traditional autoregressive token prediction, modern LLMs are increasingly steering towards *thinking* capabilities. That is, models that generate sequence of intermediate reasoning token steps, designed to tackle tasks requiring multi-step logic. A plethora of such models has emerged in this space, such as OpenAI's recent open-source GPT-OSS:120/20B variants (OpenAI 2025a). LLMs equipped with these *thinking* capabilities have shown substantially improved performance on **direct-planning** benchmarks (Valmeekam et al. 2024; Kokel et al. 2025). Although much of current research focuses on direct planning capabilities, relatively few studies have explored the application of reasoning-enabled LLMs on planning specifications. Of particular interest in our work is the LLM's potential to reason over PDDL domains. Related work is discussed in Section 7.

3 ALPSBench

ALPSBench is motivated by the lack of novel, high-quality domains for evaluating the generative capabilities of LLMs. Broadly, LLM-driven planning domain construction faces two central challenges: 1) generating correct PDDL domain specifications, and 2) detecting and repairing their specification errors. The former concerns whether LLMs can reliably generate valid domains from natural language descriptions, while the latter examines their ability to identify and correct action-semantic errors. These challenges inform the design of both ALPSGEN and ALPSQA, as detailed in Figure 1. Constructing these resources first required curating a robust suite of original and rigorously designed PDDL domains, not found in the data LLMs have been trained on.

3.1 Domains

Domain Requirements: It is critical to ensure a high-quality dataset of domains. ALPSBench follows *six* requirements. First, the following three are inspired by the filtering standards in (Hu et al. 2025): *(I)* ***Validation***: Each domain must pass syntactic validation using a PDDL parser, for which we use the PDDL toolkit[2]. *(II)* ***Complexity Control***: Domains must remain within practical limits, contain-

[2]https://github.com/AI-Planning/pddl

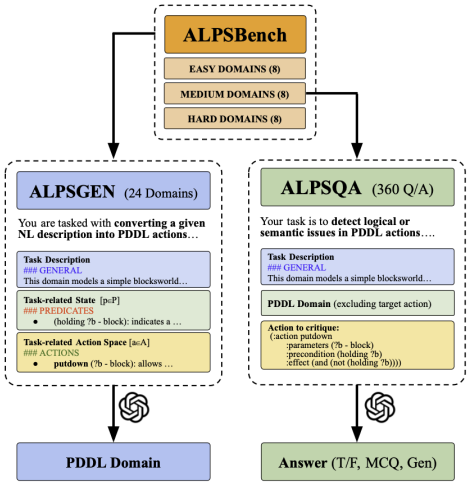

Figure 1: High-level overview of the ALPSBench benchmark, including: ALPSGEN and ALPSQA.

ing no more than 40 predicates and lifted 20 actions. Furthermore, excessively large domains make it impractical to execute meaningful plans that fully exercise the dynamics of the environment for evaluation. *(III)* ***Token Length Filtering***: Domains must not exceed 5,000 tokens as measured using the GPT-2 tokenizer[3] to respect model context window constraints. Although LLMs in our experiment are not provided with full ground-truth domains during generation, later stages (i.e., error correction) necessitate working with similarly sized representations. Beyond these standards, we introduced additional requirements tailored to ALPSBench: *(IV)* ***Clear Terminology***: Types, predicates, and actions must use clear, interpretable terminology, avoiding ambiguous or cryptic naming (e.g., Blocksworld-Mystery). *(V)* ***Plan Executability***: Each domain must be accompanied by PDDL problem files capable of producing valid, executable plans. *(VI)* ***Domain Novelty***: Domains must contain distinct planning dynamics; simply renaming ("costuming") does not alter structural similarity. However, universal terms (e.g., PICK-UP or HANDEMPTY) are permitted.

We constructed ALPSBench as a collection of 24 handcrafted classical planning domains (Table 1), grouped into three difficulty tiers with eight domains per tier. For more concrete examples, refer to Appendix C.

1. **Easy**: Domains in this category use basic :STRIPS and :TYPING with :NEGATIVE-PRECONDITIONS. Actions follow a linear pipeline with additive state transitions, and type hierarchies remain shallow (1–2 levels). The action and predicate dynamics are simple, characteristic of many well-known classical planning benchmarks.
2. **Medium**: Same PDDL requirements as *Easy*, but are more structurally complex. These domains feature (i) non-linear, branching states that require interleaved subgoal planning; (ii) consumable resources and mutually exclusive state constraints; (iii) spatial reasoning such as 3D rotation tracking, grid-based path planning, topology navigation; and (iv) deep type hierarchies (3–4 levels). Formalizing these environments required extensive, iterative refinement by human annotators.
3. **Hard**: Introduces ADL features (:ADL), enabling emergent state dynamics and non-local action effects that cannot be expressed in STRIPS. These constructs challenge LLMs by requiring reasoning about context-dependent outcomes (e.g., WHEN clauses), universal quantification over objects (FORALL), and disjunctive state constraints that, at this point, are rarely seen in training corpora.

[3]https://huggingface.co/docs/transformers/en/model_doc/gpt2

3.2 ALPSGEN

In this section, we evaluate the ability of LLMs to generate PDDL domains from natural language descriptions. Specifically, we aim to answer the following research questions:

- **RQ1:** Can LLMs infer action schemas from minimal descriptions, and how does their performance compare between full NL specifications and general action names?
- **RQ2:** Are LLMs currently capable of successfully implementing more advanced PDDL features?

Data Annotation For each domain in ALPSBench, we manually create the corresponding natural language descriptions that form our dataset, which we call *ALPSGEN*. Formally, let a domain be described by a 4-tuple: $\mathcal{I}_{\text{NL}} = \langle D_{NL}, T_{NL}, P_{NL}, A_{NL} \rangle$, where D_{NL} is the high-level summary of the domain, and T_{NL}, P_{NL}, and A_{NL} are the set of type, predicate, and action descriptions, respectively. While this level of context may seem restrictive to LLM generation, it is a necessity for ground-truth evaluation to prediction.

Our evaluation centers on the action schemas, as they form the backbone of PDDL domain specification. Therefore, we focus on exploring variability in the natural language descriptions of actions, while keeping all other components of the domain description fixed. We adopt the action-prompting strategy from Oswald et al. (2024) and generate three set variants of action descriptions:

- **Base** $N_b(A)$: Provides only the action name, its parameters, and a one-line NL summary of what the action does without explicitly mentioning any predicates.
- **Flipped** $N_f(A)$: This extends the *Base* description by additionally describing all predicates that are *deleted* preconditions of the action. For an action schema a, the description enumerates the predicates in $pre(a) \cap del(a)$.
- **Skeleton** $N_s(A)$: (our variant) A minimalist representation where we omit any NL explanation of the action; only the action's name and parameter list are given. The intention is to explore how much structural context the LLM can infer solely from the action signature.

3.3 ALPSQA

While research on LLMs for automated planning has largely focused on generating PDDL (Tantakoun, Muise, and Zhu 2025), a critical gap remains in understanding their qualitative reasoning process during generation. Our work addresses this by conducting a fine-grained evaluation of

LLMs' reasoning capabilities in detecting and repairing action-level specifications. Specifically, we aim to answer the following research questions:

- **RQ3:** Can LLMs reliably diagnose and correct erroneous PDDL actions?
- **RQ4:** Which types of PDDL action errors pose the greatest challenge for LLMs to fix?

Data Annotation To gain quantitative insights, we adopt *ACPBench* (Kokel et al. 2025) structured questionnaire framework. This comprises boolean (BOOL), multiple-choice (MCQ), and generation (GEN) format questions. In this dataset, we select the 8 domains from the *Medium* category for mutation. The mutations are formatted as:

1. **ADD** $M_{\text{add}}(a)$: A syntactically valid and unique spurious predicate p^* is inserted into the action schema $a \in A$:

$$M_{\text{add}}(a) = \begin{cases} (\text{pre}(a) \cup \{p^*\},\ \text{eff}(a)), \\ (\text{pre}(a),\ \text{eff}^+(a) \cup \{p^*\},\ \text{eff}^-(a)), \\ (\text{pre}(a),\ \text{eff}^+(a),\ \text{eff}^-(a) \cup \{p^*\}) \end{cases}$$

2. **DEL** $M_{\text{del}}(a)$: A valid predicate p is removed from either the precondition or effect of an action schema $a \in A$:

$$M_{\text{del}}(a) = \begin{cases} (\text{pre}(a) \setminus \{p\},\ \text{eff}(a)), \\ (\text{pre}(a),\ \text{eff}(a) \setminus \{p\}) \end{cases}$$

3. **FLIP** $M_{\text{flip}}(a)$: A variation of $M_{\text{add}}(a)$, where a predicate $p \in \text{pre}(a)$, or conversely, $\neg p \in \text{pre}(a)$, is incorrectly introduced into the effects $\text{eff}(a)$ with its polarity flipped:

$$M_{\text{flip}}(a):\ \text{eff}(a) \leftarrow \text{eff}(a) \cup \{\neg p\} \quad \text{where } p \in \text{pre}(a)$$

4. **RAND** $M_{\text{rand}}(a)$: The LLM must identify whether any semantic error is present in the action, without being told the fault type. For instance, instead of asking "*is this action missing a <precondition | effect> that would prevent it from being executed incorrectly?*", we ask "*is the predicate usage in this action correct?*" A mutation is sampled uniformly from the set of previous variants (1-3).

Each query to the LLM is represented by the 5-tuple: $Q = \langle \tau, D_{\text{NL}}, D_{\text{PDDL}\setminus a}, a', \mathcal{Q} \rangle$, where τ is the task instruction specifying the required JSON output, D_{NL} is the description of the domain, $D_{\text{PDDL}\setminus a}$ is the PDDL domain with the original action a removed, a' is either the original action a or a mutated variant produced by $M_\star(a)$, and $\mathcal{Q}$ is the diagnostic question associated with the mutation type $M_\star$. In total, we generated 120 unique problems, each in three question formats, for a total of 360 questions. Question type details can be found in Appendix C.3.

3.4 Preventing Benchmark Leakage

A practical concern with publicly releasing benchmarks is **training leakage**. Even when current datasets are low-contaminated (Hu et al. 2025), it does not prevent them from being incorporated into future LLM pretraining. While ACPBench (Kokel et al. 2025) outlines several mitigation strategies, these approaches demand substantial operational overhead. To address this issue, we propose a three-part framework for protecting scalable benchmark integrity:

- *First*, we propose a "Publish-Submit" model. In our case, the NL descriptions used to prompt generation are made **public**, while the ground-truth PDDL domains remain **private**. Researchers can test their systems on the public inputs, and then submit their predictions to an evaluation endpoint. The endpoint automatically runs the same evaluation pipeline used in our experiments (see Sec. 4.2) and returns the performance scores to the user.
- *Second*, to reduce the risk of overfitting, we advocate periodically introducing new domains, in the spirit of IPC benchmark cycles. This idea is consistent with prior work on *dynamic* benchmarks (Sculley et al. 2025). PlanBench (Valmeekam et al. 2023), among others, have similarly argued for systematically growing benchmarks.
- *Third*, we plan to embed watermarking signals in our domains to help detect potential training exposure. When new models are evaluated, we will apply the n-gram contamination test used in Hu et al. (2025), following the procedure described by Touvron et al. (2023) (Section A.6), to determine acceptance into the suite.

With respect to the first recommendation, active development is currently underway, and we anticipate launching the public endpoint in the near future.

4 Experimental and Evaluation Setup

4.1 Experimental Setup

Evaluated Models: Our study evaluated a range of 9 LLMs spanning 3 major model families: *o4-mini*, *o3-mini* (OpenAI 2025b), *gpt-5-mini*, *gpt-oss:120b*, *gpt-oss:20b* (OpenAI 2025a); *deepseek-r1*, *deepseek-v3* (DeepSeek-AI 2024); and *mixtral:8x22b* and *mixtral:8x7b* (Mistral 2023). All proprietary models were interfaced through their vendor-provided APIs, while open-source models were run locally on NVIDIA A40 GPUs with 128GB memory via the OLLAMA interface.[4] For consistency across all evaluations, we set the decoding temperature to $t = 0.0$, budget of thinking tokens to 8192, context window to 16k and maximum completion tokens to 8k. For ALPSGen experiments, we use a chain-of-thought (CoT) prompting strategy (Wei et al. 2022) in a zero-shot setting. For ALPSQA, we evaluate under both zero-shot and few-shot CoT settings, using examples from the BLOCKSWORLD and GRIPPERS domains.

ALPSGEN Pipelines: We explore three pipelines for action-schema construction on our ALPSGEN dataset. Each experiment retains the domain's skeletal structure while removing the actions to be generated. For reproducibility, all pipelines are implemented using **L2P** (Tantakoun, Muise, and Zhu 2025) to replicate and standardize the approaches.

- **NL2PDDL**: Following the action-by-action paradigm introduced by Guan et al. (2023) and formalized in Oswald et al. (2024), *NL2PDDL* constructs each action schema independently in sequential order.
- **ALPS-ALL**: Inspired by holistic generation (Hu et al. 2025), this approach generates the entire set of PDDL actions for a domain in a single pass.

[4] https://ollama.com

- **ALPS-ITER**: Our framework (Algorithm 1) also adopts an action-by-action process; however, instead of generating each action, the LLM is provided with all previously generated actions and can revise them when necessary.

All pipelines apply a maximum of three correction loops using the PDDL syntax checker. Feedback for iterations is derived directly from the syntax checker's error messages. Refer to Appendix C.1 for more details on error correction.

Algorithm 1: ALPS-ITER Pipeline

Input: General domain description d; NL types and predicate descriptions; NL action descriptions $\{Na_1, \ldots, Na_n\}$
Output: PDDL domain action set $\mathcal{A}$

Initialize $\mathcal{A} \leftarrow [\,]$ {Current list of generated actions}
for $k = 1$ **to** n **do**
  $Na_k \leftarrow$ NL action description for action k
  $\tilde{A} \leftarrow \mathcal{A}$ {Copy of previously generated actions}
  $\hat{a}_k, \hat{A}_{\text{rev}} \leftarrow$ LLMGENERATEORREVISE$(Na_k, \tilde{A})$ {LLM generates a new action and potential revisions to previous actions}
  $\mathcal{A} \leftarrow$ UPDATEACTIONLIST$(\mathcal{A}, \hat{a}_k, \hat{A}_{\text{rev}})$
end for
return $\mathcal{A}$

4.2 Evaluation Metrics

ALPSGEN: To assess **structural similarity** between generated and ground-truth PDDL domains, we adopt the multi-gate metric evaluation scheme introduced by Hu et al. (2025). Executability (passing the PDDL parser) serves as a gate: only parseable domains are evaluated on macro-averaged F1 scores across action parameters, preconditions, and effects. Textual similarity via Levenshtein Ratio is computed for all domains as a fallback signal. For **semantic evaluations**, we adopt the Heuristic Domain Equivalence (HDE) metric from (Oswald et al. 2024). Specifically, a generated domain D_{gen} is considered *heuristically equivalent* to the ground-truth domain D_{gt} if all plans produced by D_{gen} are valid and can be applied in D_{gt} (vice versa). We also opted for the K^* planner (Lee, Katz, and Sohrabi 2023), using 2 problem files for each domain to generate the top 100 plans, and use VAL (Howey, Long, and Fox 2004) to test for plan validity. Further details can be found in Appendix A.1.

ALPSQA: For evaluating LLMs on question-answering action schemas, we adopt straightforward correctness-based scoring tailored to each question type. Let a_{pred} denote the model's answer and a_{gold} as the ground-truth; correctness is defined as $a_{\text{pred}} = a_{\text{gold}}$ under the criteria described as:

1. **Boolean Questions (BOOL)**: Requires selecting whether an action specification is *correct* or *incorrect*.
2. **Multiple-Choice Questions (MCQ)**: Requires selecting the correct option among k choices, one of which corresponds to a "No mistakes" option.
3. **Generative Questions (GEN)**: Requires determining not only whether an error is present but also which grounded predicate is responsible. This means a prediction is correct only when it identifies the existence of an error and pinpoints its exact source predicate.

5 Experimental Results

5.1 ALPSGEN Findings

Structural Fidelity: Table 2 presents the top-performing LLMs by model family on structural similarity. There are three key findings from this table. First, **state-of-the-art LLMs achieve near-perfect executability** on *Easy* and *Medium* categories, whereas *Hard* category exhibits the worst executable performance. Consider the base descriptions: the average executability across all LLMs on ALPS-ITER is 94.44%, 88.89%, 70.83% for easy, medium, and hard, respectively; ALPS-ALL achieves 88.89%, 90.28%, 66.61%; and NL2PDDL achieves 95.83%, 90.28%, 72.22%. When combined with a feedback loop using syntax validator output messages, LLMs are highly effective at correcting mistakes. Second, **LLMs perform better at specifying action effects than preconditions**. For instance, the O4-MINI (NL2PDDL) variant shows improvements of +8.5%, +9.4%, and +20.5% on the skeleton, base, and flipped descriptions, respectively, when comparing F1 scores for effects to those for preconditions. This trend likely arises because effects are generally more directly implied within action descriptions compared to preconditions. Third, ***Skeleton* descriptions compares on-par with, or in some cases better than, the *Base* description variant**. For instance, Table 2 shows that although the skeleton variant on DEEPSEEK-R1 (ALPS-ITER) achieves slightly lower executability (-3.7%) than base, it achieves greater structural performance: +4.16% in F1-Params, +6.87% in F1-Prec, and +5.34% in F1-Eff.

Semantic Fidelity: While NL2PDDL performs comparably in structure similarity, Figure 2 shows that ALPS-ITER consistently produces more equivalent domains across all description variants. In particular, GPT-5-MINI exhibits the highest number of equivalent domains (4/16) on both the *Base* and *Skeleton* variants. Surprisingly, ALPS-ALL, with GPT-5-MINI, produced the highest number of equivalent domains (5/16) for the *skeleton* variant. Note that most models are capable of producing domains that are executable in a plan. All HDE tables can be found in Appendix D.1.

Further Analysis: Since the K^* planner currently does not support ADL PDDL features, we were unable to apply the HDE metric to the *Hard* domains. Instead, we propose a closely related evaluation procedure, which we refer to as *Partial-Heuristic Domain Equivalence* (PHDE). We construct k problem instances for each domain and generate plans using both the ground-truth domain D_{gt} and the LLM-generated domain D_{gen}, to perform cross-validation. This yields the possible classes of semantic discrepancies:

- **No Plan**: problem instance cannot be solved using D_{gen}.
- **Incompatible**: plans produced in neither domain validate when applied to the other domain.
- **Over-Restrictive**: only the plan generated from D_{gen} validates in D_{gt}, indicating that D_{gen} is too restrictive.
- **Over-Permissive**: only the plan generated from D_{gt} validates in D_{gen}, indicating that D_{gen} is too permissive.

In our experiments, we generated 4 problem files per domain and ran the FastDownward Planner (Helmert 2011)

Family	Version	EXEC. ↑			$F1_{PARAM}$ ↑			$F1_{PREC}$ ↑			$F1_{EFF}$ ↑		
		S	B	F	S	B	F	S	B	F	S	B	F
OpenAI	gpt-5-mini (NL2PDDL)	95.80	**100.0**	**100.0**	83.30	83.30	**100.0**	64.60	**66.20**	74.5	72.0	73.20	95.0
	gpt-5-mini (ALPS-A)	95.80	95.80	**100.0**	83.20	82.90	89.30	65.50	65.0	70.10	**72.80**	**74.0**	86.50
	gpt-5-mini (ALPS-I)	**100.0**	**100.0**	**100.0**	**83.33**	**83.33**	**100.0**	**65.47**	66.03	**80.60**	71.53	71.83	**95.85**
	o4-mini (NL2PDDL)	**100.0**	**100.0**	**100.0**	**86.80**	**87.50**	**100.0**	**70.60**	**68.90**	74.50	**79.10**	**78.30**	95.0
	o4-mini (ALPS-A)	95.80	95.80	93.80	82.40	80.10	88.50	64.10	65.10	68.90	73.20	72.80	84.80
	o4-mini (ALPS-I)	91.67	**100.0**	**100.0**	84.73	84.73	98.95	68.13	68.37	**77.35**	76.40	75.20	**95.45**
DeepSeek	deepseek-r1 (NL2PDDL)	**100.0**	**91.70**	**100.0**	83.30	79.20	93.80	**69.0**	64.10	72.60	74.90	72.20	89.10
	deepseek-r1 (ALPS-A)	87.50	**91.70**	**100.0**	70.70	**83.30**	93.50	61.30	**64.70**	75.10	68.30	**73.90**	89.20
	deepseek-r1 (ALPS-I)	83.80	87.50	**100.0**	**83.33**	79.17	**98.45**	68.50	61.63	**81.15**	**75.57**	70.23	**94.45**
Mistral	mixtral:8x22b (NL2PDDL)	83.30	62.50	93.75	74.50	58.10	81.25	50.90	42.80	59.90	46.70	46.70	71.50
	mixtral:8x22b (ALPS-A)	83.30	62.50	**100.0**	74.50	58.1	93.75	50.90	42.8	68.1	46.70	46.7	79.1
	mixtral:8x22b (ALPS-I)	**91.67**	**75.0**	**100.0**	**86.27**	**74.67**	**99.8**	**59.73**	**54.2**	**71.35**	**56.07**	**56.07**	**86.85**

Table 2: Top ALPSGEN results for each model family across pipelines. Metrics are reported for the S (*Skeleton*), B (*Base*), and F (*Flipped*) description variants. Best overall scores are highlighted, and **bold-faced** indicates the best variant for each model. Full table of all pipeline model performances can be found in Appendix D.1

.

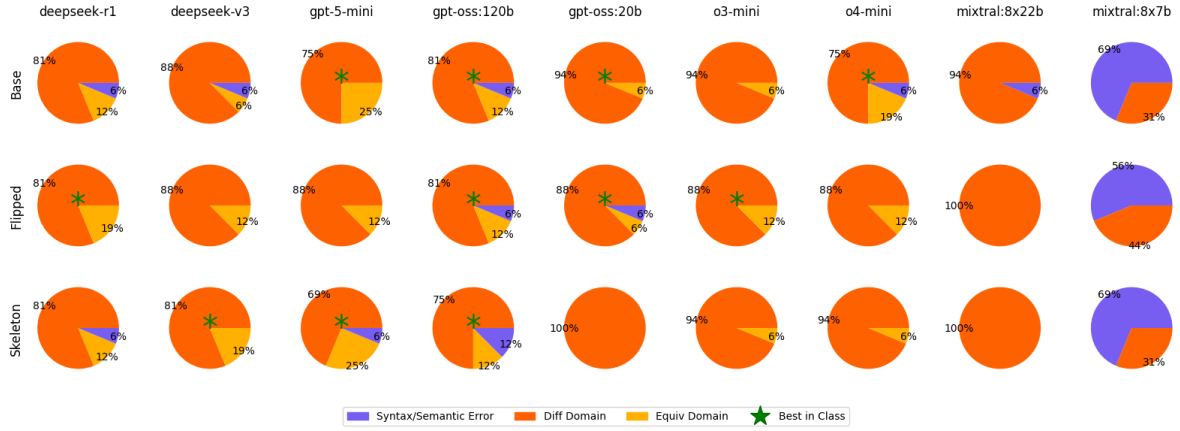

Figure 2: ALPSGEN results for HDE metric on plan executability on ALPS-ITER with 16 domains from Easy and Medium. See Appendix D for HDE results on NL2PDDL and ALPS-ALL.

configured with A^* search, equipped with the context-enhanced additive (CEA) heuristic. Figure 3 presents the results of the PHDE-based plan executability analysis. We observe that GPT-5-MINI achieves the highest overall rate of successful plan execution across all problem instances, whereas GPT-OSS:120B attains the strongest performance in terms of mutually plan-validated domains (34.4%). Across all models, the most prevalent discrepancy category is *Over Permissive*. This trend suggests that, although many generated domains can successfully execute at least one plan, they often do so by underspecifying action preconditions.

5.2 ALPSQA Findings

Table 3 displays the accuracy of LLMs across ALPSQA dataset. Three results are immediately apparent from this. First, **few-shot prompting provides a consistent performance benefit**, surpassing zero-shot prompting on average across all evaluated models, categories, and question types. This insight is most apparent in ADD, DEL, and FLIP, where the LLM is asked a targeted question about the action. Second, **LLMs tend to over-predict errors in actions**. While boolean questions generally achieve higher performance compared to MCQ and GEN questions, a noticeable level of misclassifications occurred. Table 4 shows that false positives occur in 17.5% and 18.15% of cases for zero-shot and few-shot prompting, respectively, whereas false negatives are much less frequent, at 4.63% and 4.91%. Third, **deletion predicates are harder to detect than added ones**. Specifically, the DEL mutation category consistently yields the lowest accuracy across all question formats. As shown in Table 3, average performance on DEL questions lag behind that of ADD and FLIP categories, particularly in generative settings where models must identify the missing predicate. For instance, in the zero-shot setting, DEL questions see a notable drop of -12.5% in BOOL and -10.4% in GEN accuracy relative to ADD.

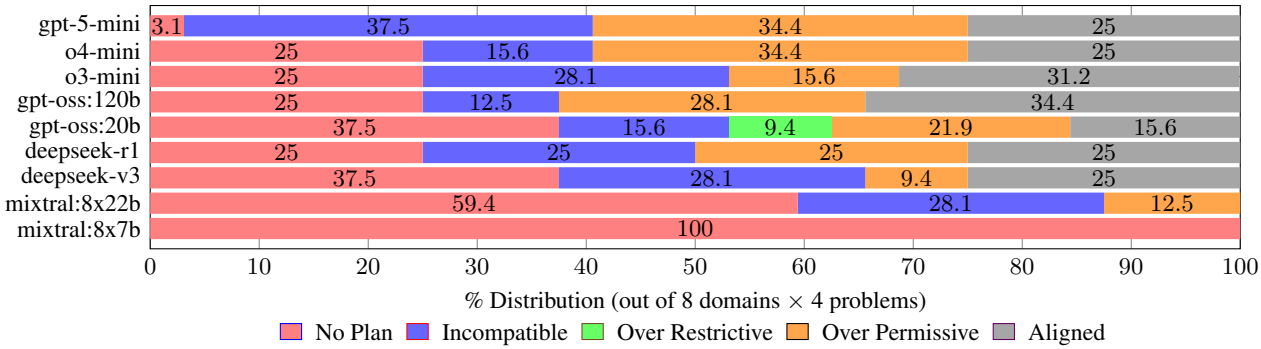

Figure 3: Distribution of PHDE evaluation on Skeleton description variant on the Hard category of ALPSGEN (8 domains).

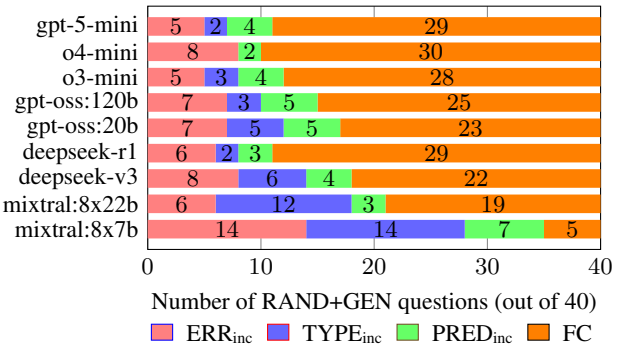

Figure 4: Performance of *RAND+GEN* in zero-shot setting. F=*Fully Correct* signifies successfully satisfying all criteria.

Further Analysis: From an application perspective, asking highly specific questions is often impractical. More general yet informative questions are preferable, motivating our RAND+GEN evaluation. In this experiment, an LLM must correctly identify: (1) whether an error exists (**ERR**); (2) the error type (i.e., *added precondition*, *missing precondition*, *added effect*, or *missing effect* (**TYPE**)); and (3) the specific grounded predicate responsible (**PRED**). As shown in Figure 4, O4-MINI (75%), GPT-5-MINI (72.5%), and DEEPSEEK-R1 (72.5%) achieve the highest accuracy across all criteria. The distribution gap between *Fully Correct* and other error modes suggests that current LLMs can somewhat reason about the specific causes of semantic errors.

6 Discussion

6.1 Revisiting the Research Questions

RQ1: Our evaluation indicates that modern LLMs can effectively infer action schemas even from highly minimal descriptions, as evidenced by their strong performance on the Skeleton variant. This demonstrates a robust capacity to generalize from action names and parameter signatures alone, leveraging built-in semantic priors about common domain dynamics. For instance, actions `light-fire` (in *Campfire*) or `pick-up-package` (in *RoboDelivery*) carry strong semantic priors that the models successfully exploit.

RQ2: Interestingly, even smaller models such as MIXTRAL:8X7B (released in December 2023) were able to generate somewhat coherent CONDITIONAL-EFFECTS and EXISTENTIAL-QUANTIFIERS. Only the strongest models could successfully apply feedback corrections to produce executable domains. These findings suggest that, given the multitude of ways to model a PDDL domain, explicitly hinting at formalism (e.g., explicitly supplying :REQUIREMENTS or using keywords like "for all" in prompts) is essential for eliciting correct and natural use of ADL constructs.

RQ3: While LLMs are generally effective at repairing PDDL syntax errors, they are far less reliable in correcting semantic errors. Across all 9 LLMs, with a total of 1080 responses for the BOOL question type, they achieved a 76.94% true positive/negative success rate in the few-shot setting, with 18.15% false positives and 4.91% false negatives. This indicates that LLMs exhibit a strong bias toward flagging issues. Notably, the RAND+GEN variant achieves only a performance accuracy of 58.3% and 63.6% in the zero-shot and few-shot setting, respectively. Even for state-of-the-art models, LLMs remain insufficiently reliable for correcting action schema errors without some sort of external feedback.

RQ4: From Table 3, we observe that missing predicates, whether in the preconditions or effects of an action, pose a significant challenge for LLMs to diagnose. With the zero-shot setting, both the BOOL and GEN question variants show substantial performance drops of -12.5% and -10.4%, respectively, when evaluating missing-predicate questions compared to their ADD counterparts. This asymmetry suggests that diagnosing under-specification remains a critical weakness in their reasoning about action semantics.

Practical Benchmark Extensions Modelling *difficulty* in PDDL domains is inherently nuanced and context-dependent. In practice, a planning domain that human engineers consider trivial may prove highly complex for an LLM to generate, depending on how the descriptions align with their pretraining, and vice-versa. We present ALPSBench as a foundational open framework and invite the community to contribute new PDDL domains. Continuously expanding the benchmark mitigates test-set overfitting, as well as progressing towards high-expressivity formalisms introduced in PDDL 2.1, 2.2, 3.0, and PDDL+.

Models	ADD ↑			DEL ↑			FLIP ↑			RAND ↑		
	Bool	MCQ	Gen	Bool	MCQ	Gen	Bool	MCQ	Gen	Bool	MCQ	Gen
GPT-5-MINI*	84.4	59.4	50.0	65.6	78.1	46.9	87.5	**81.2**	62.5	77.5	75.0	72.5
	87.5	65.6	56.2	71.9	**87.5**	50.0	**93.8**	**81.2**	68.8	67.5	**77.5**	77.5
O4-MINI*	87.5	78.1	59.4	65.6	84.4	59.4	**93.8**	68.8	62.5	90.0	72.5	**75.0**
	93.8	71.9	56.2	81.2	90.6	59.4	**93.8**	**81.2**	56.2	77.5	**77.5**	**75.0**
O3-MINI*	96.9	75.0	50.0	81.2	84.4	46.9	81.2	68.8	50.0	70.0	80.0	70.0
	87.5	84.4	53.1	**84.4**	84.4	53.1	**93.8**	75.0	56.2	67.5	**77.5**	**75.0**
GPT-OSS:120B*	81.2	**87.5**	68.8	**84.4**	90.6	43.8	**93.8**	87.5	56.2	70.0	**77.5**	62.5
	93.8	93.8	75.0	87.5	84.4	65.6	87.5	87.5	62.5	65.0	75.0	**75.0**
GPT-OSS:20B*	87.5	93.8	56.2	87.5	90.6	40.6	**93.8**	87.5	46.2	72.5	75.0	57.5
	84.4	84.4	56.2	87.5	90.6	53.1	**93.8**	75.0	81.2	70.0	70.0	72.5
DEEPSEEK-R1*	90.6	84.4	68.8	81.2	84.4	**68.8**	**93.8**	87.5	**75.0**	77.5	80.0	72.5
	93.8	**87.5**	**71.9**	78.1	**87.5**	75.0	87.5	87.5	**75.0**	77.5	**77.5**	**75.0**
DEEPSEEK-V3	50.0	65.6	43.8	50.0	78.1	40.6	56.2	62.5	50.0	**80.0**	52.5	55.0
	53.1	62.5	40.6	50.0	78.1	43.8	56.2	56.2	50.0	75.0	57.5	52.5
MIXTRAL:8X22B	84.4	43.8	62.5	56.2	46.9	34.4	100.0	50.0	62.5	72.5	30.0	47.5
	87.5	65.6	65.6	75.0	59.4	62.5	**93.8**	62.5	**75.0**	67.5	32.5	52.5
MIXTRAL:8X7B	84.4	40.6	34.4	62.5	40.6	18.8	68.8	18.8	18.8	**80.0**	20.0	12.5
	71.9	37.5	18.8	59.4	43.8	25.0	75.0	37.5	37.5	57.5	27.5	17.5
Avg. Across All Models	83.0	69.8	54.9	70.5	75.3	44.5	85.4	68.1	53.7	76.7	62.5	58.3
	83.7	72.6	54.8	75.0	78.5	54.2	86.1	71.5	62.4	69.4	63.6	63.6

Table 3: Accuracy of 9 LLMs on ALPSQA across mutation and question formats. Starred (*) LLMs indicate reasoning-oriented architecture. Best results are highlighted, second best are **boldfaced**. All models were evaluated w/ zero (top value) and two in-context examples (bottom value), within each cell.

	Predicted	
	Positive	**Negative**
Actual Positive	382 | 379	50 | 53
Actual Negative	189 | 196	459 | 452

Table 4: Zero-shot (left) and Few-shot (right) confusion matrix on Boolean questions, aggregated across all LLMs and question variants.

7 Related Work

LLM-PDDL Benchmarks: PDDL problem benchmarks (Zuo et al. 2024; Bohnet et al. 2024) constitute a fundamentally different evaluation setting than PDDL domains. In problem generation, instances vary in size, objects, and goals within the same domain, enabling automated creation of large, diverse datasets. By contrast, domain benchmarks are harder to scale: the domain is fixed, limiting both dataset size and variance in evaluation. Oswald et al. (2024), Huang, Lipovetzky, and Cohn (2024), and Gestrin, Kuhlmann, and Seipp (2024) each introduce a small number of novel domains in their evaluations, while Hu et al. (2025) address the limited size of existing datasets by proposing web-scraping PDDL domains. In our work, we introduce 24 hand-crafted novel domains designed to balance dataset size with quality. Like IPC benchmarks (Katz et al. 2025), our benchmark is designed for continuous expansion with new domains to support ongoing LLM evaluation.

LLM Detection and Repair: Gragera and Pozanco (2023) investigate whether LLMs can identify flawed PDDL domain action effects and PDDL problem initial states. Patil (2024) uses traditional error-checking methods over whole PDDL domains and provides error-message outputs as feedback to the LLM. Meanwhile, Gestrin, Kuhlmann, and Seipp (2024) propose using LLMs to evaluate its own generated specifications. Our work takes on a different approach of investigating their diagnostic capabilities of action definitions; not on the syntax level, but whether they can detect semantic inconsistencies through quantitative means.

8 Conclusion

Leveraging LLMs for modelling PDDL domains has gained significant traction in Automated Planning. Yet, it remains unclear whether they can reliably capture the underlying dynamics required for semantically grounded domain specifications. To support rigorous and forward-looking evaluation, we introduce ALPSBench, a benchmark of 24 novel, hand-crafted PDDL domains. In addition, ALPSBench contributes two branching datasets: ALPSGEN, which supports natural language descriptions for LLM-generation, and ALPSQA, a structured Q&A suite designed to probe an LLM's ability to diagnose PDDL action semantic errors. Together, these resources offer a comprehensive foundation for assessing the strengths and limitations of LLMs in planning-domain construction. We further encourage the planning community to extend ALPSBench by continually contributing new domains that advance the rigorous evaluation of LLM-based formalization.

References

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

DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. *CoRR*, abs/2412.19437.

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

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

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

Helmert, M. 2011. The Fast Downward Planning System. *CoRR*, abs/1109.6051.

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

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

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

Katz, M.; Kokel, H.; Muise, C.; Sohrabi, S.; and Sreedharan, S. 2025. Make Planning Research Rigorous Again! *CoRR*, abs/2505.21674.

Kokel, H.; Katz, M.; Srinivas, K.; and Sohrabi, S. 2025. ACPBench Hard: Unrestrained Reasoning about Action, Change, and Planning. *CoRR*, abs/2503.24378.

Lee, J.; Katz, M.; and Sohrabi, S. 2023. On K* Search for Top-K Planning. In Barták, R.; Ruml, W.; and Salzman, O., eds., *Sixteenth International Symposium on Combinatorial Search, SOCS 2023, Prague, Czech Republic, July 14-16, 2023*, 38–46. AAAI Press.

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. *CoRR*, abs/2304.11477.

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

Mistral. 2023. Mixtral of experts.

OpenAI. 2025a. gpt-oss-120b & gpt-oss-20b Model Card. *CoRR*, abs/2508.10925.

OpenAI. 2025b. OpenAI o3 and o4-mini System Card.

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

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

Sculley, D.; et al. 2025. Position: AI Competitions Provide the Gold Standard for Empirical Rigor in GenAI Evaluation. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 - Position Paper Track*. OpenReview.net.

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

Tantakoun, M.; Muise, C.; and Zhu, X. 2025. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, 25167–25188. Association for Computational Linguistics.

Touvron, H.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*, abs/2307.09288.

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

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

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

Yu, Z.; Yuan, Y.; Xiao, T. Z.; Xia, F. F.; Fu, J.; Zhang, G.; Lin, G.; and Liu, W. 2025. Generating Symbolic World Models via Test-time Scaling of Large Language Models. *Trans. Mach. Learn. Res.*, 2025.

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

A Detailed Metrics

A.1 ALPSGEN Metrics

We evaluate LLM-generated PDDL domains using the TEXT2WORLD multi-metric system, denoted as T2W-MCE, and NL2PDDL *Heuristic Domain Equivalence* (HDE) metric.

T2W-MCE Similarly, NL2PDDL follows the similar evaluation approach for action structure called Action Reconstruction Error (ARE). T2W-MCE evaluates generated domains along three complementary dimensions: **(i) *Executability* (EXEC.)**, which checks whether the predicted domain is syntactically valid and parsable; **(ii) *Structural Similarity* (SIM.)**, which measures normalized Levenshtein-based textual similarity between the generated and ground-truth PDDL; and **(iii) *Component-wise F1***, a fine-grained semantic metric applied only when the prediction is executable (EXEC. $=$ 1). For this third dimension, let D_{gt} and D_{gen} denote the ground-truth and generated domains. We compute macro-averaged F1 scores across predicates ($\mathbf{F1}_{\text{PRED}}$), action parameters ($\mathbf{F1}_{\text{PAR}}$), preconditions ($\mathbf{F1}_{\text{PRE}}$), and effects ($\mathbf{F1}_{\text{EFF}}$). *In our evaluations, we chose to exclude $\mathbf{F1}_{PRED}$ since the predicates are automatically parsed in the domain.*

Levenshtein Ratio To assess the overall structural similarity between a generated domain and its reference counterpart, we compute the Levenshtein ratio:

$$\text{Levenshtein Ratio} = 1 - \frac{d(a,b)}{|a| + |b|} \tag{1}$$

where $d(a,b)$ is the Levenshtein distance between strings a and b, and $|a|$, $|b|$ are their respective lengths. This normalized measure ranges from 0 to 1, where higher values indicate greater string similarity. If the file is executable, that is, does not contain any syntax errors, the following metrics are used to conduct further evaluation of the domain:

Action Component-Level F1 Score To capture the accuracy of individual components within each action schema—namely, parameters, preconditions, and effects—we compute component-specific F1 scores. For each component $c \in \{\text{params}, \text{preconds}, \text{effect}\}$, the average F1 score over all actions is:

$$F1_c = \frac{1}{N} \sum_{i=1}^{N} F1_{c,i} \tag{2}$$

where N is the number of actions, and $F1_{c,i}$ denotes the F1 score for component c in action i, computed as:

$$F1_{c,i} = 2 \cdot \frac{\text{precision}_{c,i} \cdot \text{recall}_{c,i}}{\text{precision}_{c,i} + \text{recall}_{c,i}} \tag{3}$$

with precision and recall defined over the predicted and reference component sets $P_{c,i}$ and $R_{c,i}$, respectively:

$$\text{precision}_{c,i} = \frac{|P_{c,i} \cap R_{c,i}|}{|P_{c,i}|}, \quad \text{recall}_{c,i} = \frac{|P_{c,i} \cap R_{c,i}|}{|R_{c,i}|}$$

For semantic evaluations, we adopt the **Heuristic Domain Equivalence** (HDE) metric. Specifically, a generated domain D_{gen} is considered *heuristically equivalent* to the ground-truth domain D_{gt} if D_{gen} is syntactically valid, and plans that are valid in D_{gt} can be applied in D_{gen}, and vice versa. Formally, for an executable prediction D_{gen}, the component-wise F1 metric is defined as:

HDE Formally, let Π_{gt} be a plan set generated from D_{gt} on some problem set, and similarly Π_{gen} for D_{gen}. Then:

$$\text{HDE}(D_{\text{gt}}, D_{\text{gen}}) = \begin{cases} 1, & \text{if } \forall \pi \in \Pi_{\text{gt}},\ \text{valid}(D_{\text{gen}}, \pi) \\ & \quad \wedge\ \forall \pi' \in \Pi_{\text{gen}},\ \text{valid}(D_{\text{gt}}, \pi'), \\ 0, & \text{otherwise.} \end{cases}$$

B ALPSGEN Evaluated Pipelines

Below are the algorithms of each pipeline's PDDL action generation strategy. We opted to run the syntax validation loop once all actions were generated, as implemented in TEXT2WORLD. First, *ALPS-ITER* (Algorithm 1) follows an action-by-action generation. At each action, the LLM is also prompted with previous actions, as well as the option to revise previous actions to replace in the list. Second, *ALPS-ALL* (Algorithm 2) generates all the actions in a single pass—given the whole action descriptions. Third, *NL2PDDL* (Algorithm 3) follows an action-by-action strategy, but each prompt only contains the isolated action description. That is, the actions are generated individually in a single pass.

Algorithm 2: ALPS-ALL Pipeline

Input: General domain description d; NL types and predicate descriptions; NL action descriptions $\{Na_1, \ldots, Na_n\}$
Output: PDDL domain D_{PDDL}

Initialize $\mathcal{A} \leftarrow [\,]$ {Current list of generated actions}
// Generate actions all at once
$\mathcal{A} \leftarrow \text{LLMGENERATEALL}\{Na_1, \ldots, Na_n\}$
$D_{\text{PDDL}} \leftarrow (\mathcal{A}$, :requirements, :types, :predicates)
// Final domain validation loop
while D_{PDDL} fails syntax validation **do**
    $D_{\text{PDDL}} \leftarrow$ LLM(retry, error msg.)
end while
return D_{PDDL}

Algorithm 3: NL2PDDL Pipeline

Input: General domain description d; NL types and predicate descriptions; NL action descriptions $\{Na_1, \ldots, Na_n\}$
Output: PDDL domain D_{PDDL}

Initialize $\mathcal{A} \leftarrow [\,]$ {Current list of generated actions}
for $k = 1$ **to** n **do**
    $Na_k \leftarrow$ NL action description for action k
    $\hat{a}_k \leftarrow \text{LLMGENERATE}(Na_k)$
    $\mathcal{A} \leftarrow \text{APPENDACTIONLIST}(\hat{a}_k)$
end for
$D_{\text{PDDL}} \leftarrow (\mathcal{A}$, :requirements, :types, :predicates)
// Final domain validation loop
while D_{PDDL} fails syntax validation **do**
    $D_{\text{PDDL}} \leftarrow$ LLM(retry, error msg.)
end while
return D_{PDDL}

C Prompting

C.1 ALPSGEN Error Correction Prompt:

The following prompt was used to correct errors in the PDDL domain based on the syntax error message returned by the PDDL checker:

````
I would like you to serve as an expert in PDDL,
assisting me in correcting erroneous PDDL code. I will
provide you with the incorrect PDDL along with the
error messages returned by the system. You should
output your thought process firstly. You MUST enclose
the COMPLETE corrected PDDL within ```pddl```.

Hints to help you debug the pddl domain file:
1. You should start by checking if all the essential
domain constructs or domain definition constructs are
present. Commonly included domains comprise:
    a. :domain declaration to name the domain.
    b. :requirements to specify the PDDL features
        used in the domain.
    c. :types to define any object types for
        categorizing entities in the planning problem.
    d. :constants (if necessary) to declare any
        objects that remain static throughout.
    e. :predicates to define the properties and
        relations between objects that can change.
    f. :functions (if necessary) to define numeric
        functions for more complex domains.
    g. :action definitions for each action that
        agents can perform, including parameters,
        preconditions, and effects.
2. You need to check the number of parameters
    of each actions.
3. Having :typing in the domain indicates that it
    uses a hierarchy to organize objects. Therefore,
    it's crucial to clearly list all object types
    related to the planning task in a :types section.
4. '-' should not appear in :types.

Incorrect PDDL:
```pddl
  1 | (define (domain logistics-strips)
  2 |     (:requirements :strips :typing)
  3 | ...
 23 |     (:action LOAD-TRUCK
 24 |        :parameters (?obj - obj ?truck - truck
                ?loc - location)
 25 |        :precondition
 26 |           (and
 27 |               (OBJ ?obj)
 28 |               (TRUCK ?truck)
 29 |               (LOCATION ?loc)
 30 |               (at ?obj ?loc)
 31 |               (at ?truck ?loc))
 32 | ...
```
Error Information:
PDDLValidationError: Type mismatch in argument 1 of
predicate 'at' found in :precondition of action 'LOAD-
TRUCK': got `?truck` assigned type `truck`, expected:
    * `obj` (or its subtype(s))
Parsed error line: (at ?truck ?loc)
````

C.2 ALPSGEN Prompting

To illustrate our NL descriptions for ALPSGEN, we show an example of the FIREFIGHTER domain (Figure 5). For each domain, we supply their descriptions for LLM generation. Figure 6 demonstrates an example of the description variants (Skeleton, Base, and Flipped).

C.3 ALPSQA Prompting

Table 5 shows the specific format for each question types (Bool, MCQ, and Gen). Below is an example of a complete prompt given to an LLM for MCQ (ADD):

````
You are an expert in automated planning and PDDL
semantics. Your task is to detect logical or
semantic issues in PDDL actions with respect to
a domain description. You must select the most
**correct option** from the choices below that
describes whether the action contains an
inconsistency that would compromise the behavior
of the overall domain.

You must provide your final answer strictly in
**valid JSON** format:
{{
  "answer": "A", "B", "C", "D",
  "reason": "<one-sentence explanation>"
}}

### DOMAIN DESCRIPTION
```
{domain_desc}
```

### PDDL DOMAIN:
```
{domain_pddl}
```

### ACTION TO EVALUATE
Action: {action_name}
Action Description: {action_desc}
Action PDDL:
```
{action_pddl}
```

**Question**: Given the description and PDDL domain,
identify the precondition which could prevent the
action from being executed correctly.
Choose one option only: A, B, C, D.

**Choices**:
{choices}

Put your answer here:
{{
  "answer": "",
  "reason": ""
}}
````

```
### GENERAL
This domain models a rescue scenario where firefighters navigate hallways and rooms to clear debris and rescue
civilians. Rooms and hallways that contain fire cannot be extinguished and debris block firefighters' movements.
Firefighters can move between connected locations, enter or exit rooms, carry debris to clear way, and carry
people to safety. Additionally, once a firefighter carries a debris, they can only move if they place it back
down. Firefighters can go to a location with debris, but they cannot move away from it until it is picked up.

### TYPES
- **location - object**: a position in the building that contains people or objects. It is a subtype of **object**
- **person - object**: an individual, either a person or firefighter. It is a subtype of **object**
- **hallway - location**: a passage connecting rooms or other hallways where firefighters can move. It is
    a subtype of **location**
- **room - location**: enclosed space connected to hallways. It is a subtype of **location**
### PREDICATES
- **(carrying-debris ?p - person)**: true if firefighter <?p> is currently carrying a debris.
- **(carrying-person ?p - person)**: true if firefighter <?p> is currently holding a person.
- **(connected ?h1 - hallway ?h2 - hallway)**: true if hallway <?h1> is connected to hallway <?h2>.
- **(debris-at ?l - location)**: true if debris is present at location <?l>.
- **(fire-at ?l - location)**: true if fire is located at location <?l>.
- **(holding ?p1 - person ?p2 - person)**: true if firefighter <?p1> is holding civilian <?p2>.
- **(is-civilian ?p - person)**: true if person <?p> is a civilian.
- **(is-firefighter ?p - person)**: true if person <?p> is a firefighter.
- **(person-at ?p - person ?loc - location)**: true if person <?p> is currently at location <?loc>.
- **(room-at ?r - room ?h - hallway)**: true if room <?r> is accessible from hallway <?h>.
```

Figure 5: General description for domain FIREFIGHTER.

```
### SKELETON
- **pickup-debris (?p - person ?loc - location)**: description N/A

### BASE
- **pickup-debris (?p - person ?loc - location)**: The action **pickup-debris** will have a firefighter
    lift a debris at location <?loc>.

### FLIPPED
- **pickup-debris (?p - person ?loc - location)**: The action **pickup-debris** will have a firefighter lift a
    debris at location <?loc> if location <?loc> contains debris and person <?p> is not currently carrying debris.
```

Figure 6: Natural language description variants for action PICKUP-DEBRIS in FIREFIGHTER domain.

	BOOL (T/F)	**MCQ (Choose: A, B, C, ...)**	**GEN (Predicate: <predicate> | "None")**
ADD	...does this action **include** a <*precondition* | *effect*> that would prevent it from being executed?	...identify the **added** <*precondition* | *effect*> which could prevent the action from being executed correctly.	...identify the **added** <*precondition* | *effect*> which could prevent the action from being executed correctly.
DEL	...is this action **missing** a <*precondition* | *effect*> that would prevent it from being executed incorrectly?	...identify the **missing** <*precondition* | *effect*> that would prevent the action from being executed incorrectly.	...identify the **missing** <*precondition* | *effect*> that would prevent the action from being executed incorrectly.
FLIP	...does this action **include** a predicate that has been flipped, causing states to behave incorrectly when executed?	...identify the predicate in the action that has been flipped, causing the states to behave incorrectly when executed.	...identify the predicate in the action that has been flipped, causing the states to behave incorrectly when executed.
RAND	...is the predicate usage in this action correct?	...what is the most **probable error** in this action regarding its predicate usage?	...what is the most **probable error** in this action regarding its predicate usage?

Table 5: ALPSQA Question Taxonomy: Four perturbation types (ADD, DEL, FLIP, RAND) across three question formats.

D Detailed Results

D.1 ALPSGEN Extra Results

Figures 7, 8, and 9 show the full evaluation results for the Heuristic Domain Equivalence metric. Tables 6, 7, and 8 show the full results for T2W-MCE metrics for all models across description variants (Skeleton, Base, and Flipped).

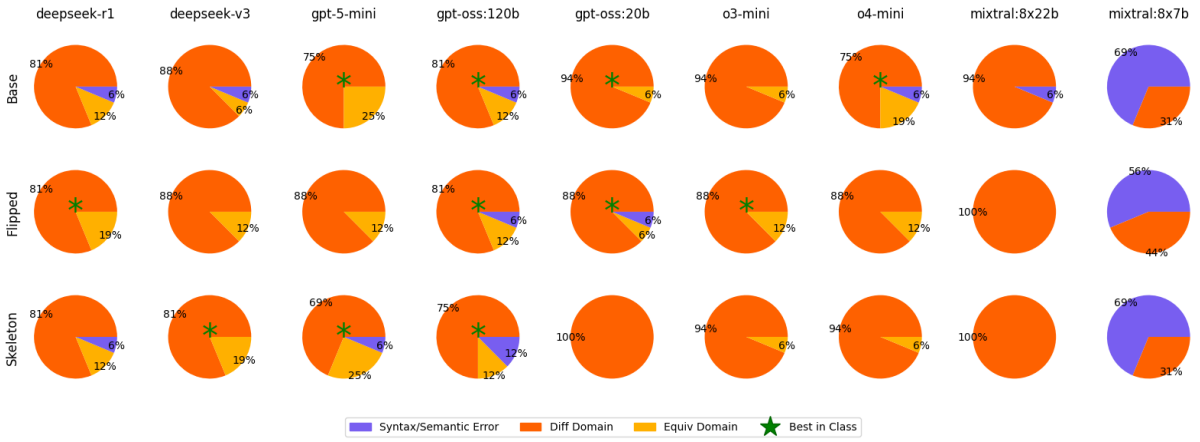

Figure 7: ALPSGEN results for HDE metric on plan executability on ALPS-ITER

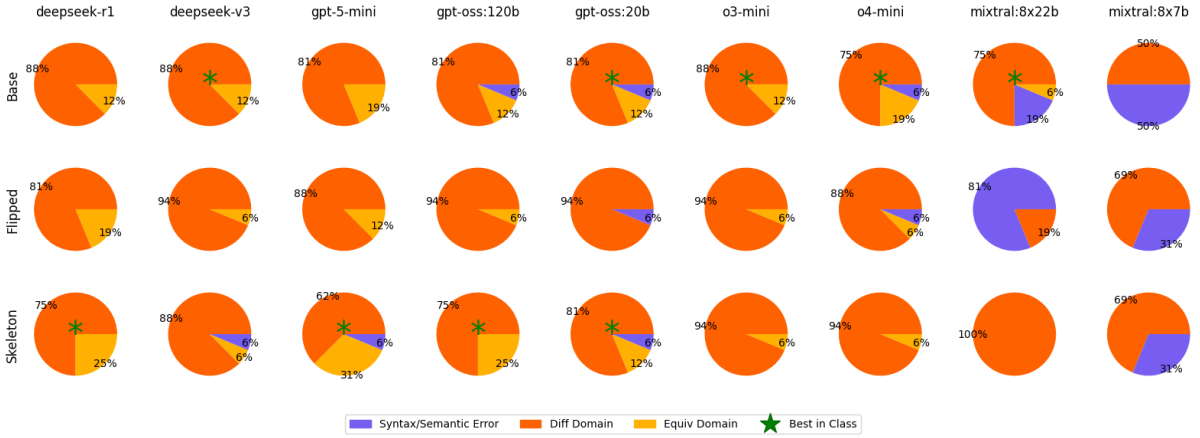

Figure 8: ALPSGEN results for HDE metric on plan executability on ALPS-ALL

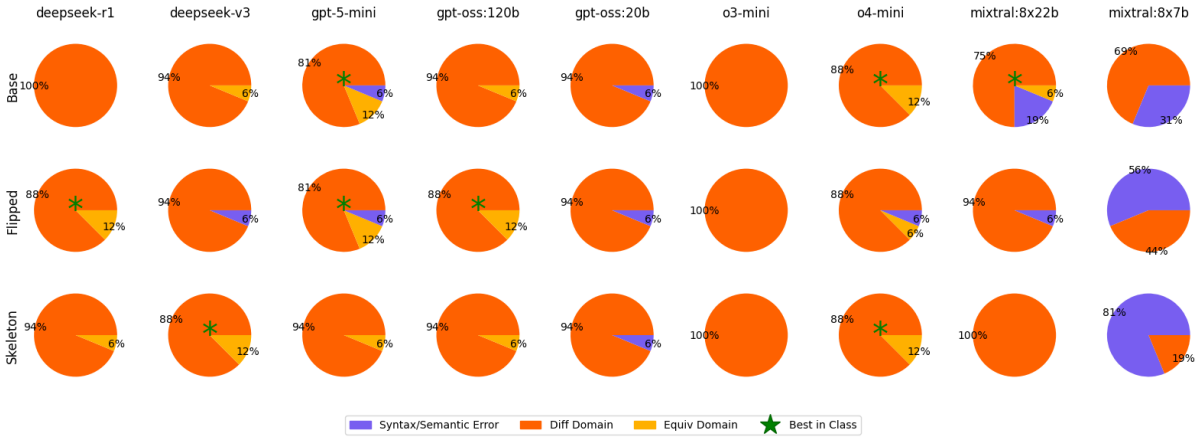

Figure 9: ALPSGEN results for HDE metric on plan executability on NL2PDDL

ALPS-ITER

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	**83.3**	**100.0**	85.2	87.0	70.8	83.3	57.8	65.5	62.8	71.5
	O4-MINI	33.3	91.7	81.3	87.0	33.3	84.7	27.5	68.1	29.8	76.4
	O3-MINI	79.2	95.8	84.9	86.6	70.8	**87.5**	54.2	66.8	62.0	75.6
	GPT-OSS:120B	79.2	95.8	85.7	87.3	**74.9**	87.4	57.4	67.6	**67.1**	**77.5**
	GPT-OSS:20B	75.0	87.5	83.7	78.8	54.2	62.4	40.3	47.2	47.3	54.6
DeepSeek	DEEPSEEK-R1	75.0	83.3	**87.1**	**89.2**	70.8	83.3	**58.9**	**68.5**	65.0	75.6
	DEEPSEEK-V3	75.0	91.7	85.4	86.6	66.6	78.9	53.7	63.4	58.0	70.0
Mistral	MIXTRAL:8X22B	62.5	91.7	81.9	84.5	45.8	86.3	45.8	59.7	49.5	66.8
	MIXTRAL:8X7B	16.7	20.8	63.3	61.0	16.7	16.7	11.1	11.1	10.0	10.0
	AVG.	64.4	84.3	82.0	83.1	57.9	74.5	45.2	57.6	50.2	64.2

ALPS-ALL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	79.2	**95.8**	85.4	86.9	**75.0**	**83.2**	**61.4**	**65.5**	67.6	72.8
	O4-MINI	83.3	**95.8**	86.5	**87.5**	74.7	82.4	61.0	64.1	66.8	**73.2**
	O3-MINI	79.2	**95.8**	86.1	86.9	**75.0**	79.2	57.7	61.4	66.1	68.3
	GPT-OSS:120B	**87.5**	91.7	**86.7**	87.1	74.7	78.9	60.0	62.5	**68.2**	72.1
	GPT-OSS:20B	79.2	87.5	81.7	79.4	58.2	66.6	43.7	50.5	47.7	55.4
DeepSeek	DEEPSEEK-R1	62.5	87.5	82.7	86.1	54.2	70.7	48.8	61.3	52.8	68.3
	DEEPSEEK-V3	70.8	91.7	85.0	87.0	62.5	82.7	49.3	63.6	51.5	69.0
Mistral	MIXTRAL:8X22B	66.7	83.3	81.7	83.1	62.5	74.5	42.6	50.9	49.7	58.8
	MIXTRAL:8X7B	29.2	41.7	70.7	72.5	8.3	20.8	4.7	13.3	6.0	15.0
	AVG.	71.3	85.64	82.94	84.06	60.57	71.0	47.69	54.79	52.93	61.43

NL2PDDL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	83.3	95.8	85.6	87.0	75.0	83.3	58.5	64.6	64.9	72.0
	O4-MINI	**95.8**	**100.0**	**87.7**	**88.1**	**82.6**	86.8	**67.8**	**70.6**	**75.4**	**79.1**
	O3-MINI	83.3	**100.0**	85.4	86.8	75.0	87.3	54.9	63.6	63.0	73.6
	GPT-OSS:120B	75.0	91.7	85.1	86.5	75.0	**87.5**	56.3	64.9	64.3	76.0
	GPT-OSS:20B	54.2	87.5	83.4	80.3	50.0	74.9	40.0	57.0	43.8	65.2
DeepSeek	DEEPSEEK-R1	83.3	**100.0**	86.4	**88.1**	75.0	83.3	62.5	69.0	67.3	74.9
	DEEPSEEK-V3	16.7	95.8	77.5	86.6	16.7	83.0	14.6	65.0	15.7	74.3
Mistral	MIXTRAL:8X22B	66.7	83.3	81.7	83.1	62.5	74.5	42.6	50.9	49.7	58.8
	MIXTRAL:8X7B	4.2	12.5	73.1	65.9	4.2	12.5	2.5	6.7	3.1	8.7
	AVG.	62.5	85.18	82.88	83.6	57.33	74.79	44.41	56.92	49.69	64.73

Table 6: Performance comparison of different LLMs across different pipelines for **SKELETON** descriptions. EC_k denotes a setting permitting K correction steps by the model. (EC_0: zero-shot w/o correction, EC_3: w/ 3 correction steps). **Bold-faced** values indicate the best performance within each metric category. Grey-highlighted values indicate the best overall performance across all pipelines.

ALPS-ITER

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	79.2	**100.0**	**85.7**	87.5	**75.0**	83.3	59.8	66.0	66.1	71.8
	O4-MINI	33.3	**100.0**	81.8	**89.0**	26.4	84.7	19.2	**68.4**	22.9	**75.2**
	O3-MINI	**87.5**	**100.0**	85.4	87.1	83.3	**87.5**	**60.9**	63.9	**70.7**	74.6
	GPT-OSS:120B	62.5	91.7	84.1	87.3	58.3	83.3	43.8	64.3	51.9	74.1
	GPT-OSS:20B	70.8	95.8	84.0	83.3	54.2	70.7	44.4	55.7	49.2	61.6
DeepSeek	DEEPSEEK-R1	66.7	87.5	**85.7**	84.1	62.5	79.2	49.6	61.6	55.7	70.2
	DEEPSEEK-V3	62.5	91.7	84.7	83.9	58.3	79.0	48.1	62.1	51.4	68.7
Mistral	MIXTRAL:8X22B	58.3	75.0	81.6	83.4	58.3	74.7	42.6	54.2	43.0	56.1
	MIXTRAL:8X7B	8.3	20.8	64.3	67.2	8.3	16.7	6.5	10.1	6.3	11.4
	AVG.	58.8	84.7	81.9	83.6	53.9	73.2	41.7	56.3	46.4	62.6

ALPS-ALL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	**91.7**	**95.8**	87.2	87.3	**78.9**	82.9	**62.4**	**65.0**	**70.1**	**74.0**
	O4-MINI	79.2	**95.8**	86.3	87.8	67.9	80.1	55.7	65.1	62.0	72.8
	O3-MINI	**91.7**	**95.8**	**87.0**	87.7	75.0	79.2	59.6	61.9	66.9	69.9
	GPT-OSS:120B	79.2	87.5	86.4	87.7	70.7	78.9	56.0	61.3	62.7	69.9
	GPT-OSS:20B	75.0	83.3	82.1	76.4	62.4	70.2	47.3	52.1	53.6	60.6
DeepSeek	DEEPSEEK-R1	62.5	91.7	84.4	**88.4**	58.3	**83.3**	48.8	64.7	53.2	73.9
	DEEPSEEK-V3	62.5	91.7	83.8	86.5	58.3	74.7	46.9	60.1	49.9	64.8
Mistral	MIXTRAL:8X22B	45.8	62.5	80.6	81.5	45.8	58.1	33.8	42.8	36.9	46.7
	MIXTRAL:8X7B	20.8	33.3	75.4	75.1	12.5	24.7	9.9	18.7	7.8	14.9
	AVG.	67.6	81.9	83.7	84.3	58.9	70.2	46.7	54.6	51.5	60.8

NL2PDDL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	79.2	**100.0**	85.5	87.6	75.0	83.3	59.1	66.2	67.4	73.2
	O4-MINI	**87.5**	**100.0**	**87.0**	**88.3**	**79.2**	**87.5**	**63.2**	**68.9**	**70.3**	**78.3**
	O3-MINI	79.2	**100.0**	84.9	87.4	75.0	**87.5**	56.5	63.5	65.6	75.6
	GPT-OSS:120B	75.0	95.8	85.0	87.6	66.7	87.3	50.8	65.4	57.9	76.4
	GPT-OSS:20B	66.7	87.5	84.4	77.2	62.5	75.0	46.6	57.1	51.3	63.3
DeepSeek	DEEPSEEK-R1	79.2	91.7	86.9	88.6	70.8	79.2	57.9	64.1	64.1	72.2
	DEEPSEEK-V3	50.0	**100.0**	82.0	87.4	45.8	83.1	38.7	63.0	39.2	71.5
Mistral	MIXTRAL:8X22B	45.8	62.5	80.6	81.5	45.8	58.1	33.8	42.8	36.9	46.7
	MIXTRAL:8X7B	4.2	37.5	73.5	72.1	4.2	33.3	3.3	19.8	2.8	22.7
	AVG.	63.0	86.1	83.3	84.2	58.3	74.9	45.5	56.8	50.6	64.4

Table 7: Performance comparison of different LLMs across different pipelines for **BASE** descriptions.

ALPS-ITER

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	87.5	**100.0**	88.2	89.7	**87.5**	**100.0**	**73.6**	80.6	**84.6**	**95.9**
	O4-MINI	25.0	**100.0**	81.6	89.9	23.9	99.0	20.8	77.4	24.0	95.5
	O3-MINI	87.5	87.5	87.3	87.2	**87.5**	87.5	67.7	67.7	83.5	83.5
	GPT-OSS:120B	81.3	**100.0**	87.1	89.3	79.6	98.4	64.2	**82.6**	74.6	92.1
	GPT-OSS:20B	87.5	**100.0**	85.9	88.2	62.5	75.0	50.4	60.7	59.2	70.1
DeepSeek	DEEPSEEK-R1	87.5	**100.0**	**89.4**	**90.8**	86.0	98.5	71.6	81.2	82.9	94.5
	DEEPSEEK-V3	**93.8**	**100.0**	88.1	88.9	86.8	93.0	70.2	74.7	79.3	85.1
Mistral	MIXTRAL:8X22B	87.5	**100.0**	87.2	88.8	**87.5**	99.8	64.6	71.4	76.4	86.9
	MIXTRAL:8X7B	25.0	37.5	66.8	73.0	25.0	37.5	20.4	27.9	19.4	28.4
	AVG.	73.6	91.7	84.6	87.3	69.6	87.6	55.9	69.3	64.9	81.3

ALPS-ALL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	**93.8**	**100.0**	88.1	89.1	83.4	89.3	66.5	70.1	81.3	86.5
	O4-MINI	75.0	93.8	86.3	88.8	70.2	88.5	56.7	68.9	66.7	84.8
	O3-MINI	**93.8**	93.8	**88.3**	88.2	**93.8**	**93.8**	**71.1**	71.1	**89.7**	89.7
	GPT-OSS:120B	87.5	**100.0**	87.9	89.7	81.1	93.6	65.3	73.9	79.9	**91.1**
	GPT-OSS:20B	81.3	93.8	86.8	87.9	74.9	80.9	58.6	63.3	69.0	74.4
DeepSeek	DEEPSEEK-R1	87.5	**100.0**	88.2	**89.8**	81.3	93.5	67.5	75.1	77.8	89.2
	DEEPSEEK-V3	87.5	**100.0**	87.4	89.2	81.3	**93.8**	67.9	**75.4**	77.0	88.1
Mistral	MIXTRAL:8X22B	**93.8**	**100.0**	86.7	87.8	87.5	**93.8**	63.7	68.1	73.3	79.1
	MIXTRAL:8X7B	37.5	56.3	78.6	80.5	31.3	50.0	24.5	35.6	18.6	33.3
	AVG.	82.0	93.1	86.5	87.9	76.1	86.4	60.2	66.8	70.4	79.6

NL2PDDL

Family	Version	EXEC. ↑		SIM. ↑		$F1_{PARAM}$ ↑		$F1_{PREC}$ ↑		$F1_{EFF}$ ↑	
		EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3	EC_0	EC_3
OpenAI	GPT-5-MINI	87.5	**100.0**	87.1	88.8	87.5	**100.0**	68.8	74.5	83.7	95.0
	O4-MINI	**93.8**	**100.0**	88.6	89.7	**93.8**	**100.0**	**71.2**	74.5	**89.6**	**95.4**
	O3-MINI	**93.8**	**100.0**	87.5	88.5	**93.8**	**100.0**	64.0	66.8	87.8	93.4
	GPT-OSS:120B	87.5	**100.0**	86.9	88.7	87.5	**100.0**	66.1	71.3	83.0	94.0
	GPT-OSS:20B	75.0	**100.0**	85.2	87.8	75.0	93.7	53.0	64.4	68.9	84.8
DeepSeek	DEEPSEEK-R1	**93.8**	**100.0**	**88.8**	**90.0**	87.5	93.8	69.4	72.6	83.6	89.1
	DEEPSEEK-V3	50.0	**100.0**	83.4	89.1	50.0	99.8	40.5	**75.7**	47.1	94.3
Mistral	MIXTRAL:8X22B	50.0	93.8	82.5	81.8	50.0	81.3	39.9	59.9	43.6	71.5
	MIXTRAL:8X7B	6.3	31.3	73.2	67.2	6.3	31.3	4.2	21.6	4.2	24.7
	AVG.	70.9	91.7	84.8	85.7	70.2	88.9	53.0	64.6	65.7	82.5

Table 8: Performance comparison of different LLMs across different pipelines for **FLIPPED** descriptions.