# ACPBench: Reasoning About Action, Change, and Planning

**Harsha Kokel, Michael Katz, Kavitha Srinivas, Shirin Sohrabi**

IBM Research
{Harsha.Kokel,Michael.Katz1,Kavitha.Srinivas}@ibm.com, ssohrab@us.ibm.com

**Abstract**

There is an increasing body of work using Large Language Models (LLMs) as agents for orchestrating workflows and making decisions in domains that require planning and multi-step reasoning. As a result, it is imperative to evaluate LLMs on core skills required for planning. In this work, we present ACPBench, a benchmark for evaluating the reasoning tasks in the field of planning. The benchmark consists of 7 reasoning tasks over 13 planning domains. The collection is constructed from planning domains described in a formal language. This allows us to synthesize problems with provably correct solutions across many tasks and domains. Further, it allows us the luxury of scale without additional human effort, i.e., many additional problems can be created automatically. Our extensive evaluation of 21 LLMs and OpenAI o1 reasoning models highlight the significant gap in the reasoning capability of the LLMs. Our findings with OpenAI o1, a multi-turn reasoning model, reveal significant gains in performance on multiple-choice questions, yet surprisingly, no notable progress is made on boolean questions.

**Dataset** — https://ibm.github.io/ACPBench

**Extended version** —
https://doi.org/10.48550/arXiv.2410.05669

## 1 Introduction

Recent research has explored the potential of using Large Language Models (LLMs) as reasoners for solving multi-step reasoning problems (Chu et al. 2024). Building on their success in certain reasoning tasks and benchmarks, there is a growing interest in using LLMs as agents for orchestrating workflows and making decisions in domains that require planning (Huang et al. 2024; Wang et al. 2024a). This is a promising area of research, with potential applications in various fields. However, there is a lack of systematic evaluation of LLMs reasoning and planning capabilities.

This work aims at evaluating and improving language models' ability to plan. However, end-to-end evaluation of planning ability is challenging. One, if an agent reaches a goal it does not necessarily mean it can plan. Second, evaluating a plan might be difficult in a domain where there can be multiple plans to achieve the goal. So, instead of focusing on the entire end-to-end planning ability, we distill 7 atomic reasoning tasks that are critical for reliable planning and create datasets of such tasks. These tasks focus on reasoning about Actions, Change (transitions) and Planning; hence, we call our benchmark as ACPBench. The tasks include single step reasoning, like evaluating whether an action can be performed in the described state, as well as multi step reasoning, like whether a sequence of actions is a valid plan for the described state and the described goal.

For each task, ACPBench features both boolean (Bool) and multiple-choice (MCQ) style questions from 13 domains. All the datasets are generated from a formal representation of the domain in Planning Domain Definition Language (PDDL) (McDermott 2000). Twelve of these domains are well-established benchmarks in both planning and reinforcement learning communities, readily available in PDDL format. Inspired by the shuffle task in BigBenchHard Suite (Suz-

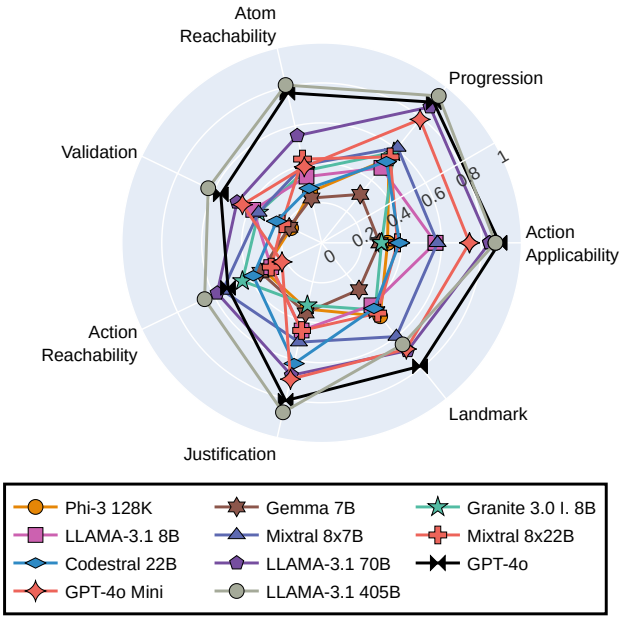



Figure 1: Performance of few state-of-the-art LLMs over different tasks in ACPBench.

gun et al. 2023), we have created an additional domain from scratch. The benefit of constructing the dataset from PDDL descriptions is twofold. First, it allows us to use existing planning tools and second, and arguably more important, it allows obtaining *provably correct* information for all the tasks. Natural language templates for these domains were carefully crafted by 5 researchers. These templates and planning tools enable us to generate massive data for each task.

We evaluate performance of OpenAI o1 reasoning model and 21 state-of-the-art language models (including open-sourced `Phi-3 128K` (Abdin et al. 2024), `Granite 3.0` (Granite Team 2024), `Mixtral 8x22B` (MistralAI 2024), `LLAMA-3 70B` (Dubey et al. 2024), and a closed source `GPT-4o` (OpenAI 2024a)) on the ACPBench. We found that, with Chain-of-Thought prompting (COT) (Wei et al. 2022) and 2-shot examples, `GPT-4o` was only able to achieve $78.40\%$ accuracy on MCQ questions in the ACPBench; with lowest accuracy of $52.50\%$ for the most difficult (validation) task. Similarly, OpenAI o1 preview achieves accuracy of $87.31\%$ on average for the MCQ questions, with lowest accuracy of $63.08\%$ for the most difficult task. Figure 1 shows the overall performance of few selected models on all 7 tasks of ACPBench. To understand whether the smaller language models can improve their performance on these tasks, we finetune a language model on these tasks. The finetuning resulted in substantial improvements in performance across tasks and even demonstrated the ability to generalize to previously unseen domains.

In summary, our contributions are as follows:

- We identify a collection of 7 reasoning tasks required for efficient planning and introduce the first of its kind large-scale benchmark—ACPBench.
- We evaluate OpenAI o1 reasoning models and 21 state-of-the-art language models of different sizes on ACPBench.
- We finetune a 8B parameter model and show that the finetuned model performs on par with the large models.
- We conduct ablation studies as follows: (a) to understand effects of in-context example and COT, (b) to investigate if tasks in ACPBench capture the plan generation ability, and (c) to understand how LLMs' abilities have progressed over time for ACPBench tasks.

## 2 Related Work and Background

Recognizing the importance of evaluating reasoning and planning ability of LLMs, various benchmarks have been proposed (Liu et al. 2024; Ma et al. 2024). Most relevant to our work are the benchmarks that are generated from PDDL tasks. He et al. (2023) proposed a natural language based question answering style dataset to evaluate LLMs on 4 tasks of projection, execution, planning, and goal recognition. PlanBench (Valmeekam et al. 2023b) is a benchmark suite with 8 planning tasks including plan generation, reasoning about plan execution, and plan verification. Both these benchmarks focus on a limited number of planning domains (mainly the BlocksWorld domain), employing a template-based approach to generate natural language text. In contrast, AutoPlanBench (Stein et al. 2024) proposes to leverage LLMs to generate the natural language template. They prompt an LLM for natural language template per predicate and per action. By reducing the human effort required for template generation, they were able to scale up the dataset to 12 domains. However, they limit their focus to a single task - plan generation.

```
Context: This is a swap domain where agents
  are swapping items or roles. Each agent
 is always assigned a single item/role. The
  goal is to obtain desired items/roles
 assigned. There are 8 agents: carol,
 michelle, xena, vic, dave, zoe, heidi, and
  alice. There are 8 items/roles:
 quadcopter, frisbee, necklace, whale,
 iceskates, guitar, zebra, and slinky.
 Currently, heidi is assigned necklace,
 michelle is assigned quadcopter, dave is
 assigned iceskates, vic is assigned whale,
  xena is assigned slinky, carol is
 assigned frisbee, alice is assigned zebra,
  and zoe is assigned guitar.
Bool: Is the following action applicable in
  this state: trade guitar of zoe for
 iceskates of dave?
MCQ: Which of the following actions will be
  applicable in this state?
A. exchange frisbee of carol with zebra of
 alice.
B. exchange guitar of zoe with necklace of
 vic.
C. exchange guitar of heidi with zebra of
 zoe.
D. exchange guitar of vic with zebra of zoe
 .
```

Figure 2: Example of boolean and multi-choice questions from the Applicablity task in ACPBench. The context contains the domain and the problem description. Query to LLM consists of context and a boolean or multi-choice question.

In parallel, Handa et al. (2024) proposed ActionReasoningBench, featuring six tasks: Fluent Tracking, State Tracking, Action Executability, Effects of Actions, Numerical RAC, and Composite Questions. Although there is some overlap between the tasks in ActionReasoningBench and ACPBench (for example, the Effects of Actions task overlaps with our Progression task), the majority of the tasks we propose are not covered by ActionReasoningBench: Reachability, Action Reachability, Validation, Justification, Landmarks. Similarly, the following ActionReasoningBench tasks are not covered in ACPBench: State Tracking, and Numerical RAC.

We now switch to providing the necessary background. The ACPBench questions collection is generated based on PDDL tasks. A PDDL task is defined over the first-order language; consisting of predicates, variables, and objects. A state $s$ is defined as a conjunction of grounded (by objects) predicates, also called atoms. An action $a$ is defined as a triple $\langle pre(a), add(a), del(a)\rangle$; consisting of preconditions, add effects and delete effects, each being a conjunction of atoms. An action $a$ is applicable in a state $s$ if the state satisfies the preconditions of the action, i.e $pre(a) \subseteq s$. On performing an action $a$ in state $s$, the world transitions to the next state $t = s[a] = s \setminus del(a) \cup add(a)$. A goal $g$ is also a conjunction of atoms, and a state $s$ is a goal state if

$g \subseteq s$. A sequence of actions $\pi_s = a_1 \dots a_n$ is applicable in the state $s$ if the actions are applicable in a sequence to the resulting states. $\pi_s$ is a plan for the state $s$ if $\pi_s$ is an applicable sequence of actions that results in a goal state.

# 3 ACPBench

## 3.1 Domains

ACPBench collection consists of 11 classical planning domains, Alfworld (Shridhar et al. 2021), and a novel swap domain. The 11 classical planning domain, which were also used by AutoPlanBench (Stein et al. 2024), have public problem instance generators (Seipp, Torralba, and Hoffmann 2022). Alfworld is a text-based reinforcement learning environment where an agent is given household tasks like 'put a pan on the table'. Alfworld uses tasks from the Alfred dataset (Shridhar et al. 2020) and encodes the dynamics of the domain in publicly available PDDL[1]. Corresponding PDDL problem files are obtained from the MINT benchmark (Wang et al. 2024b). For the novel Swap domain, we created the PDDL domain and the problem instance generator. Figure 2 contains an example problem description in this domain. All the domains are summarized in Table 1.

We meticulously curated a set of templates to transform the PDDL task into a natural language description. Following AutoPlanBench, we explored using LLMs to automatically generate the templates, however, we found the outcome not reliable and needed significant modification. So, instead, 5 researchers crafted the translation, carefully selecting and refining the templates to ensure they accurately convey the desired information. Specifically, we made templates for *domain description*, *problem description* and *actions*, from which we can compose (partial) states – current state or a goal. These three templates, together with the PDDL files, are to be provided for every new domain, should we decide to extend the benchmark in the future.

## 3.2 ACPBench Tasks

We focus on 7 reasoning tasks within the realm of planning. For each task, we provide a description and explain how the data was collected.

**Applicability (App)** The first, basic requirement for efficient planning is to determine the valid, available actions in a given situation. Various existing work have discussed LLMs fall short of this basic ability. When using GPT-4 Turbo for travel planning, Xie et al. (2024) found that more than 30% of the failed plans had *invalid action dead loop*–that is even when the model was informed that the action is invalid, LLMs repeated these actions.

For an action to be valid, its preconditions must hold in the state. Given a state $s$ and the set of actions $O$, the subset of applicable actions would be $O(s) = \{a \in O \mid pre(a) \subseteq s\}$, easily computable by iterating over the actions. We therefore can create a boolean question with a positive answer by sampling from $O(s)$ and with a negative answer by sampling from $O \setminus O(s)$. A multiple-choice question can be created by sampling the correct answer from $O(s)$ and wrong candidates from $O \setminus O(s)$. Figure 2 shows example domain and problem description used in the context as well as examples of Bool and MCQ questions for the applicability task.

[1]https://github.com/alfworld/alfworld/blob/master/alfworld/data/alfred.pddl

| Domain | # Pred. | # Actions | Max char. |
|---|---|---|---|
| Blocksworld | 5 | 4 | 1770 |
| Logistics | 9 | 6 | 1065 |
| Grippers | 4 | 3 | 1057 |
| Grid | 12 | 5 | 1235 |
| Ferry | 7 | 3 | 2132 |
| FloorTile | 10 | 7 | 3196 |
| Rovers | 25 | 9 | 3631 |
| VisitAll | 3 | 1 | 1347 |
| Depot | 6 | 5 | 1301 |
| Goldminer | 12 | 7 | 1140 |
| Satellite | 8 | 5 | 4302 |
| Swap | 1 | 1 | 849 |
| Alfworld | 34 | 19 | 4099 |

Table 1: Statistics of the 13 domains in ACPBench. The top 8 domains are used for both finetuning and evaluation. The bottom 5 domains are exclusively used for evaluations. Columns indicates the number of predicates and lifted actions in the PDDL domain, as well as the max character length of the NL problem description in the generated dataset.

**Progression (Prog)** The next task evaluates LLMs ability to understand the outcome of an action or change. This ability is important to track information across transitions. The subpar performance of LLMs on the *Tracking Shuffled Objects* task in the Big Bench Hard dataset suggests a significant limitation in their ability to reason about the consequences of actions or changes (Suzgun et al. 2023). Further, a few papers have proposed to use LLMs to execute a plan. For example, Wang et al. (2023) asks LLM to devise a plan and execute it step-by-step to reach the goal. To faithfully execute a plan, it is important for LLMs to demonstrate understanding of progression; how the world state is changed by the action.

When a valid action is performed, the state changes in the following manner: The delete effects of that action will no longer hold and the add effects will hold. Everything else remains unchanged. Given a state $s$ and an action $a$, the next state is $t = s \setminus del(a) \cup add(a)$. We can now partition the facts in the problem into four sets: the facts that held before applying the action and still hold ($s \cap t$), the facts that held before but not anymore ($s \setminus t$), those that did not hold but now hold ($t \setminus s$), and those that did not hold before and still don't hold ($F \setminus (s \cup t)$). While the answer of whether the fact is true after applying the action depends only on whether it is in $t$, the chain of thoughts leading to the answer differs for the aforementioned four cases. We construct a boolean question by sampling from each of the four fact sets (if they are not empty), getting at most two positive and two negative examples per state. A single MCQ is constructed by sampling one possible answer from each of the four fact sets (non-empty ones), according to a uniform procedure described above.

**Reachability (Reach)** The reachability task evaluates if a specific sub-goal can eventually be reached from the given state by taking (possibly multiple) actions. This is a multi-step reasoning task that can help avoid exploring unfeasible options. To maximize the efficiency of LLMs, it is crucial to detect unreachable (sub)goals early on. This can avoid unnecessary prompting and wasteful exploration, ensuring that the LLMs are utilized effectively, especially when used during search (Yao et al. 2023).

Reachability is PSPACE-hard to answer positively in general (Bylander 1994) for a specific fact, since that would require an evidence - a sequence of actions that achieves a state where the specified facts hold. However, generating positive examples is easy, based on any action sequence, taking the facts out of the end state. For negative examples, we explore multiple cases of unreachable facts and fact pairs. First, existing planning methods (under)approximate the reachability with poly-time computable delete-relaxed reachability (Hoffmann and Nebel 2001). Facts that are not delete-relaxed reachable are therefore guaranteed not to be reachable. Another possible reason for a pair of facts that are individually reachable not to be reachable in the same state is if they are mutually exclusive (Lin 2004; Fišer and Komenda 2018). A simple example of mutually exclusive facts in the ferry domain are *(empty-ferry)* and *(on ?c)*, meaning that the ferry cannot be empty and at the same time a car is on the ferry. Third, static facts that are not true in the initial state will never become true. For instance, c0 can never become a location, so *(location c0)* is unreachable (not captured by the methods in the first case, as they focus solely on non-static predicates). The chain of thoughts for a positive example is based on a sequence of actions that achieve the fact. For the negative examples, the chain of thoughts follows the argument laid out above for each of the cases. As in the previous case, the MCQ is captured by choosing from the lists of positive and negative options.

**Action Reachability (AReach)** In API-driven workflows, the objective is typically presented as an instruction to execute a specific function (Qin et al. 2024). In these scenarios, an LLM must identify the necessary prerequisites for execution and formulate a strategy to meet them. Therefore, it is essential for LLMs to assess whether a given instruction is executable from the provided starting point. We formulate this ability as action reachability task.

The action reachability task is closely related to the atom reachability. If an action model is available, then action reachability is equivalent to the atom reachability over the preconditions of the action. Therefore, this task requires an additional reasoning step about action preconditions. Similarly to the atom reachability task, the positive examples are generated from action rollouts, while the negative examples are generated by collecting actions with preconditions including unreachable atoms according to two of the three cases mentioned above delete-relaxed reachability and mutexes. The third case, unreachable static facts, was not used as often creates non-sensible actions *board car l0 at location c1*. Instead, we added incorrect action templates for each action, like *"board the car c1 at location l0 into the airplane"* or *"drive from location l0 to location l1"*. Here as well, the chain of thoughts are created in a similar manner, and the MCQ is captured based on the positive and negative options lists.

**Validation (Val)** A body of research has advocated the use of LLMs for validation and refinement (Shinn et al. 2023; Gou et al. 2024; Madaan et al. 2023). In line with this research, we propose a Validation task. Here, given an initial state and a goal condition, the objective is to assess whether the specified sequence of actions is valid, applicable, and successfully achieves the intended goal.

There are essentially only four options in this case: (a) the sequence is not valid, (b) the sequence is valid, but not applicable, (c) the sequence is valid, applicable, but does not achieve the goal, and (d) the sequence is a plan. These are the four options used for all MCQ for this task. Since the options do not change, we generate four questions per sample, for each of the options to be a correct answer. In the boolean case, we create six different questions, with positive and negative variants for the three cases of whether the sequence is valid, applicable, and a plan. We generate the data for these questions from plans as follows. For the case (c), starting from a plan, we replace a suffix with a random rollout, ensuring that the goal is not achieved at the end of the rollout, but the sequence remains applicable. For the case (b), we try to replace an action on the sequence with an inapplicable action (one whose precondition does not hold in the state), starting from the end of the sequence. Once successful, we return the sequence ending with the inapplicable action. For the case of (a), we simply randomly choose an action on the sequence to replace its template with an incorrect action template, as in the previous task.

**Justification (Just)** A major criteria for plans to be considered reasonable is whether they include unnecessary actions. In the realm of LLMs and API workflows, it is desirable to avoid calling unnecessary APIs as well as reduce wasteful explorations. Hence, it would be of immense value if LLMs are able to identify whether an action is necessary. This corresponds to the justification task in planning literature.

The justification task reasons whether every action is actually needed on the plan. The problem was studied in the literature (Fink and Yang 1992; Salerno, Fuentetaja, and Seipp 2023) and found to be NP-hard in general. However, optimal plans are known to have all their actions being justified and checking whether a single action or a pair of consequent actions can be removed can be done in polynomial time. We consider the following cases, for either a single action or a pair of consequent actions in a plan: 1) a single action can be removed from the plan and the remaining plan is still a valid plan for the same problem 2) an action cannot be removed from the plan 3) the consequent pairs of actions can be removed from the plan 4) the immediate pairs of action cannot be removed from the plan. Note that we truncate the considered plans and only consider two actions after the goal is reached except if the truncation leads to a non-plan. Given a large set of plans, we consider the above 4 cases, and generate positive and negative examples.

**Landmarks (Land)** LLMs have shown to hallucinate or deviate from the task when the trajectory is long (Huang et al. 2024). To alleviate this problem, various work has proposed to use LLMs to decompose the goal into subgoals and achieve each of these subgoals separately. To do this faithfully, it is crucial for LLMs to be able to identify subgoals that are necessary to achieve the goal. In planning literature such subgoals are often called landmarks (Porteous, Sebastia, and Hoffmann 2001). Landmarks are facts that must become true sometime along every plan. So, the last task in ACPBench evaluates LLMs ability to recognize landmarks.

While checking whether a fact is a landmark is PSPACE-hard (Porteous, Sebastia, and Hoffmann 2001), there are several methods that can find a subset of landmarks (Keyder, Richter, and Helmert 2010; Hoffmann, Porteous, and Sebastia 2004; Richter, Helmert, and Westphal 2008; Zhu and Givan 2003). We use the so-called RHW method (Richter, Helmert, and Westphal 2008). Further, negative evidence can be obtained from a collection of plans - a fact that does not appear on all of these plans is not a landmark. We sample from positive and negative examples obtained that way and construct two boolean questions and one MCQ. Here as well, the chain of thoughts generated capture the described logic.

### 3.3 Data Generation

We use 25 PDDL problem files of varying sizes per domain. These 25 tasks are partitioned into a training and a test set. For each task, we use classical planners to generate a large collection of 1000 plans (Katz and Lee 2023; Katz and Sohrabi 2020). With these plans, we sample the state space as follows. First, given a set of plans, we gather the states along these plans. Then, in order to obtain a diverse sample, we run random rollouts from each of the states found. The number of plans and the sample size are configurable parameters. In the *landmarks* task described above, we also find plans for the sampled states. To do that, we replace the initial state with the sampled state in the planning problem instance and run a top-k planner (Katz and Lee 2023). For finding mutexes, we exploit lifted mutex groups implementations from Fišer (2020). In this manner, we can potentially generate as many examples as we want. But to keep the test set of reasonable size, we generate only 10 examples per domain, per task.

## 4 Experiments

### 4.1 Evaluation of Pre-trained Language Models

We first analyse how pre-trained language models perform on ACPBench. Table 2 presents the accuracy of all the language models on the 7 ACPBench tasks. These results are mean over 5 runs for all models; except GPT family models and LLAMA-3.1 405B (Dubey et al. 2024), which were run once due to resource constraints. All LLMs were either accessed using API or hosted locally using hugging face transformer library on machines with 2 $A100$ 80 GB GPU. Note, accuracy of 50.00 on boolean questions indicates that the performance of the model is as good as a random guess. As all the MCQs in the datasets have 4 options, accuracy less than 25.00 indicates that the performance is worse than random guess. To investigate the out-of-the-box performance,

```
**Question**: This is a ferry domain, where
 the task is to transport cars from their
start to their goal locations, using a
ferry. Each location is accessible by
ferry from each other location. The cars
can be debarked or boarded, and the ferry
can carry only one car at a time.  There
are 2 locations and 2 cars, numbered
consecutively.  Currently, the ferry is at
 l0, with the car c1 on board. The cars
are at locations as follows: c0 is at l0.
 Is the following action applicable in
this state:  travel by sea from location
l1 to location l0?
**Thoughts**: Let's think step by step.
Step 1: In order to apply the action travel
 by sea from location l1 to location l0,
the following fact(s) must hold in this
state: The ferry is at l1 location.
Step 2: These facts do not hold in the
mentioned state.
So, the action is not applicable.
**Final Answer**: No.
**Question**: ...
**Thoughts**: ...
**Final Answer**: Yes.
**Question**:  <context> + <question>
**Thoughts**: Let's think step by step.
```

Figure 3: Example of the COT prompt.

we restrict the evaluation to single turn COT prompting with two in-context examples. An example prompt for the Bool applicability question is shown in Figure 3.

Notably, LLAMA-3.1 405B and GPT-4o consistently outperform other models on these tasks, although they do not **always** achieve the top performance. When it comes to smaller open-sourced models, Codestral 22B stands out for its exceptional performance on boolean questions, while Mixtral 8x7B excels in handling multi-choice questions. However, both of them lag significantly behind GPT-4o, which is the best performer in these tasks. Action Reachability and Validation are the most challenging tasks for LLMs. Surprisingly, the GPT family models are not even among top-3 for the Action Reachablity task. Across all the tasks, GPT-4o performs best for boolean and LLAMA-3.1 405B performs best for multi-choice questions.

Figure 4 displays a domain-wise analysis of the performance of larger LLMs on multi-choice questions. The average performance of these top-5 models is shown in Figure 4 as the dotted line in black. This indicates that across models no specific domain seems too easy. However, Rovers, FloorTile, Blocksworld, Alfworld and Satellite domains pose the greatest challenges to LLMs, in that particular order.

### 4.2 Finetuning

Foundational models, and LLMs specifically, have shown to improve performance on specific tasks when they are finetuned for those tasks. So, next we investigate if finetuning a language model provides any improvement. For this investigation, we keep aside the following 5 domains, Depot, Goldminer, Satellite, Swap, and Alfworld, and generate a

| Model | Applicability | | Progression | | Reachability | | Validation | | Action Reach. | | Justification | | Landmark | | Mean | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bool | MCQ | Bool | MCQ | Bool | MCQ | Bool | MCQ | Bool | MCQ | Bool | MCQ | Bool | MCQ | Bool | MCQ |
| Phi-3 128K | 66.15 | 33.08 | 68.46 | 53.85 | 52.31 | 26.15 | 50.77 | 19.23 | 53.33 | 32.50 | 49.23 | 33.85 | 49.23 | 46.92 | 55.53 | 34.75 |
| Gemma 7B | 63.23 | 28.62 | 64.92 | 31.08 | 53.08 | 23.08 | 46.92 | 20.0 | 55.67 | 34.50 | 50.77 | 36.46 | 27.54 | 30.31 | 51.80 | 28.93 |
| Mistral 7B | 61.54 | 32.31 | 73.08 | 38.46 | 53.08 | 28.46 | 47.85 | 17.69 | **65.00** | 19.17 | 48.46 | 30.00 | 35.38 | 33.08 | 55.00 | 28.67 |
| Mistral I. 7B | 63.08 | 31.54 | 61.54 | 46.92 | 61.54 | 33.08 | 52.15 | 36.15 | 45.83 | 34.17 | 43.08 | 29.23 | 57.69 | 50.77 | 55.45 | 37.30 |
| Granite C. 8B | 59.23 | 32.31 | 70.00 | 34.31 | 52.31 | 24.31 | 44.15 | 17.08 | 57.50 | 25.83 | 46.92 | 34.62 | 37.23 | 35.38 | 53.09 | 29.21 |
| Granite 3.0 8B | 72.31 | 26.92 | 73.08 | 53.85 | 53.08 | 24.62 | 53.08 | 20.00 | 45.83 | 30.83 | 49.23 | 34.62 | 42.31 | 34.62 | 55.56 | 32.21 |
| Granite 3.0 I. 8B | 76.92 | 30.00 | 73.85 | 57.69 | 53.08 | 36.92 | 55.38 | 34.62 | 58.33 | 44.17 | 70.77 | 31.54 | 51.54 | 43.08 | 62.84 | 39.72 |
| LLAMA-3 8B | 72.92 | 49.23 | 73.08 | 56.00 | 55.23 | 41.08 | 51.54 | 49.23 | 63.50 | 36.67 | 57.54 | 32.31 | 56.92 | 43.85 | 61.53 | 44.05 |
| LLAMA-3.1 8B | 65.38 | 56.92 | 63.85 | 47.69 | 53.08 | 33.85 | 60.00 | 37.69 | 42.50 | 28.33 | 46.92 | 45.38 | 33.85 | 40.00 | 51.46 | 41.52 |
| Mixtral 8x7B | 75.85 | 57.69 | 74.00 | 61.38 | 76.00 | 40.00 | 65.69 | 34.77 | 52.83 | 55.00 | 55.38 | 51.38 | 59.54 | 60.00 | 65.53 | 51.44 |
| Codestral 22B | 84.62 | 39.23 | 83.85 | 51.54 | 54.62 | 28.46 | 66.15 | 24.62 | 53.33 | 38.33 | 67.69 | 62.31 | 59.23 | 42.31 | 67.40 | 40.97 |
| Mixtral 8x22B | 80.77 | 37.69 | 72.31 | 54.62 | 50.00 | 42.62 | 37.69 | 16.92 | 58.50 | 27.83 | 43.08 | 44.62 | 44.77 | 45.23 | 55.63 | 39.25 |
| Deepseek I. 33B | 70.77 | 37.23 | 68.46 | 46.31 | 53.08 | 31.69 | 51.54 | 37.69 | 50.00 | 27.50 | 46.92 | 26.15 | 62.31 | 39.23 | 57.58 | 35.11 |
| LLAMA C. 34B | 80.77 | 42.31 | 73.08 | 43.85 | 53.08 | 25.69 | 50.15 | 28.46 | 53.17 | 33.33 | 55.38 | 35.38 | 46.92 | 40.62 | 59.02 | 35.71 |
| LLAMA-2 70B | 78.46 | 24.62 | 71.54 | 36.77 | 53.08 | 26.92 | 51.38 | 16.15 | 60.83 | 22.00 | 49.23 | 55.54 | 24.46 | 26.00 | 55.72 | 29.71 |
| LLAMA C. 70B | 74.77 | 36.15 | 54.77 | 52.92 | 48.62 | 23.69 | 40.0 | 17.69 | 49.67 | 28.83 | 46.92 | 31.54 | 37.08 | 42.31 | 50.90 | 32.87 |
| LLAMA-3 70B | 90.77 | 82.31 | 93.08 | 86.15 | **87.69** | **82.31** | **78.62** | 56.62 | 60.50 | 63.00 | 62.31 | 85.38 | 78.15 | 64.77 | 78.71 | 74.30 |
| LLAMA-3.1 70B | 93.08 | 84.31 | 89.85 | 86.77 | 61.38 | 54.92 | 66.15 | 46.62 | 63.00 | 58.00 | 56.92 | 68.46 | 34.62 | 69.23 | 66.67 | 66.94 |
| LLAMA-3.1 405B | 95.38 | 86.92 | 93.08 | **93.85** | 59.23 | 80.77 | 77.23 | **62.92** | **65.00** | **65.00** | **90.00** | **86.92** | 83.08 | 65.38 | 80.49 | **77.42** |
| GPT-4o Mini | 90.77 | 73.85 | **95.38** | 79.23 | 80.77 | 39.23 | 67.69 | 46.15 | 54.17 | 21.67 | 77.69 | 70.00 | 76.92 | 67.69 | 77.74 | 56.50 |
| GPT-4o | **96.92** | **89.23** | 94.62 | 90.00 | 79.23 | 76.92 | 61.54 | 53.85 | 57.50 | 52.50 | 88.46 | 80.77 | **95.38** | **79.23** | **81.84** | 74.97 |

Table 2: Accuracy of 21 LLMs, (I)nstruct and (C)ode models, on 7 ACPBench tasks (boolean and multi-choice). The best results are **boldfaced**, second best are *underlined*, and the best among the small, open-sourced models are *double underlined*. All models were evaluated with two in-context examples and COT prompt. The right-most column is mean across tasks.

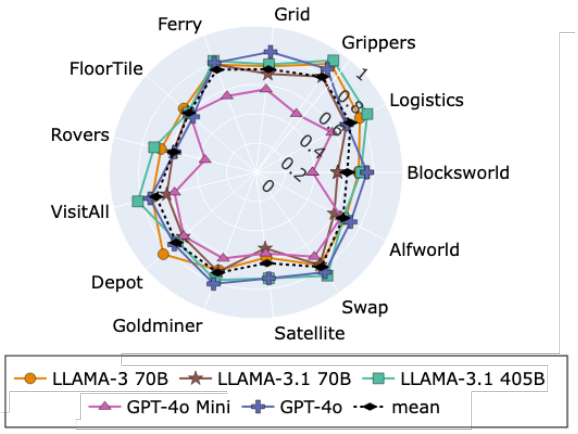



Figure 4: Comparison of 5 top performing LLMs on MCQ in 13 domains of ACPBench. The mean of performance across these models is presented with dotted line in Black. The mean line indicates that none of the domains are exceptionally easy.

training set for the remaining 8 domains. Then we pick one of the small models, Granite-code 8B (Mishra et al. 2024), and finetune it with QLoRA. The resulting performance improvement is shown in Table 3a. As finetuned models have already seen examples during training, we use only IO prompts with the finetuned model. We finetuned Granite-code 8B available on HuggingFace with two A100 80GB GPUs.

Upon finetuning, the average accuracy of the model improves from 51.43% to 95.71% on boolean questions and from 19.18% to 94.29% on multi-choice questions. Further, Table 3b presents the performance on the remaining 5 unseen domains. It is remarkable to observe such a significant improvement even on unseen domains; sometimes surpassing the GPT-4o performance. This indicates that finetuning a model, even on a separate domain, improves performance on these tasks. The right-most column in Tables 3a and 3b presents the performance of the best on that task LLM with COT 2-shots prompting. As can be seen; Granite Finetuned model outperforms the best of all models for most of the tasks in the training domains. Even in testing domains, the accuracy difference is significantly reduced upon finetuning.

### 4.3 Ablations

**Prompt Style** From previous section, it is clear that COT 2-shot yields better results than IO prompts for ACPBench tasks. However, it is not clear whether COT or 2-shot examples provide the performance gain. To investigate this, we perform the following ablation study. We compare four prompt styles: (1) IO prompt, (2) Chain-of-Thought prompt without in-context examples (COT), (3) IO prompt with two in-context examples (IO 2-shots), and (4) Chain-of-Thought with two in-context examples (COT 2-shots).[2]

We include Granite-code 8B base model, LLAMA-3 70B (one of the top-performing open source model), and the Granite finetuned model. To have a fair comparison, we use 2-shot examples from the training domains and only compare performance on the testing domains for MCQ tasks. Figure 5 presents the results. For the two pretrained models, we see that while COT 2-shots prompting yields better result than IO, IO 2-shots prompting had the best performance. For finetuned model, we see that neither COT nor 2-shots provide any advantage; rather IO prompts yield the best results.

[2]Examples of prompts are included in the extended version.

| Task | | Base IO | Base COT 2-shot | Finetuned IO | Best |
|---|---|---|---|---|---|
| App | Bool | 53.75 | 62.5 (+8.75) | **98.75** (+45.0) | 97.50 |
| | MCQ | 15.0 | 36.75 (+21.75) | **92.5** (+77.5) | 90.00 |
| Prog | Bool | 52.5 | 76.25 (+23.75) | **97.5** (+45.0) | 96.25 |
| | MCQ | 22.5 | 33.25 (+10.75) | **93.75**(+71.25) | **93.75** |
| Reach | Bool | 47.5 | 52.5 (+5.0) | **97.5** (+50.0) | 87.50 |
| | MCQ | 15.0 | 20.75 (+5.75) | **98.75**(+83.75) | 82.5 |
| Val | Bool | 45.0 | 40.5 (−4.5) | **100** (+55.0) | 78.75 |
| | MCQ | 38.5 | 20.0 (−18.5) | **87.5** (+49.0) | 57.75 |
| AReach | Bool | 45.0 | 56.25 (+11.25) | **97.5** (+52.5) | 65.75 |
| | MCQ | 14.25 | 28.75 (+14.5) | **95.0** (+80.75) | 78.75 |
| Just | Bool | 56.25 | 50.0 (−6.25) | **97.5** (+41.25) | 90.0 |
| | MCQ | 16.25 | 35.0 (+18.75) | **96.25** (+80.0) | 82.5 |
| Land | Bool | 60.0 | 41.25 (−18.75) | 81.25(+21.25) | **97.50** |
| | MCQ | 20.0 | 18.5 (−1.5) | **90.0** (+70.0) | 71.25 |
| Mean | Bool | 51.43 | 54.18 (+2.75) | **95.71**(+44.28) | 81.07 |
| | MCQ | 20.21 | 27.57 (+7.36) | **93.39**(+73.18) | 77.68 |

(a) Training

| Task | | Base IO | Base COT 2-shot | Finetuned IO | Best |
|---|---|---|---|---|---|
| App | Bool | 50.0 | 54.0 (+4.0) | 74.0 (+24.0) | **96.00** |
| | MCQ | 14.0 | 25.2 (+11.2) | 62.0 (+48.0) | **88.00** |
| Prog | Bool | 50.0 | 60.0 (+10.0) | 80.0 (+30.0) | **94.00** |
| | MCQ | 28.0 | 36.0 (+8.0) | 82.0 (+54.0) | **96.0** |
| Reach | Bool | 46.0 | 52.0 (+6.0) | 82.0 (+36.0) | **88.00** |
| | MCQ | 10.0 | 30.0 (+20.0) | 56.0 (+46.0) | **82.00** |
| Val | Bool | 46.0 | 50.0 (+4.0) | 80.0 (+34.0) | **84.0** |
| | MCQ | 26.0 | 12.4 (−13.6) | 54.0 (+28.0) | **71.2** |
| AReach | Bool | 35.0 | 60.0 (+25.0) | **82.5** (+47.5) | 77.5 |
| | MCQ | 5.0 | 20.0 (+15.0) | **70.0** (+65.0) | 57.50 |
| Just | Bool | 42.0 | 42.0 (+0.0) | **98.0** (+56.0) | 96.0 |
| | MCQ | 16.0 | 34.0 (+18.0) | 80.0 (+64.0) | **94.0** |
| Land | Bool | 44.0 | 30.8 (−13.2) | 72.0 (+28.0) | **92.0** |
| | MCQ | 20.0 | 62.4 (+42.4) | 92.0 (+72.0) | **94.0** |
| Mean | Bool | 44.71 | 49.83 (+5.21) | 81.21 (+36.5) | **82.79** |
| | MCQ | 17.00 | 31.43 (+14.43) | 70.86(+53.86) | **78.07** |

(b) Testing

Table 3: Comparison of the Granite-code 8B model (Base) to Finetuned on (a) 8 training domains and (b) 5 domains of ACPBench. Columns *Base IO* and *Base COT 2-shot*, presents the accuracy values for the Base model with Input-Output prompts (IO) and COT prompt with two in-context examples (COT 2-shot), resp. *Finetuned IO* presents model accuracy with IO prompts. The values in parentheses represent the improvement over the base model w/ IO prompts. The rightmost column (*Best*) presents the the best LLM with COT 2-shot on training and test domains, resp. Best results bolded.

**Generalization** ACPBench consists of tasks that are crucial for effective, robust and reliable planning. Improving performance on ACPBench should improve LLM's ability to reason about these tasks, and hence should improve LLM's ability to generate plans. To verify this hypothesis, we compare the Granite-code Base 8B model and Granite finetuned model on plan generation task (t1) in PlanBench (Valmeekam et al. 2023a). Table 4 presents the results,

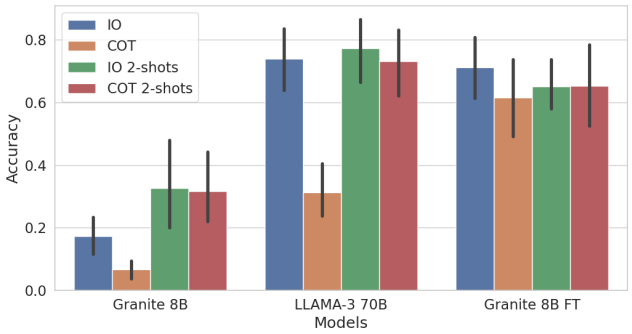



Figure 5: Comparison of different prompt styles on two pretrained models: Granite 8B and LLAMA-3 70B, and Granite finetuned model for MCQ tasks in 5 testing domains.

| Domain | Base | Finetuned | LLAMA-3 70B |
|---|---|---|---|
| Blocksworld (600) | 24 | 44 | 57 |
| Logistics (285) | 14 | 15 | 14 |

Table 4: Comparison of Granite-code Base, Finetuned, and LLAMA-3 70B model on PlanBench Dataset.

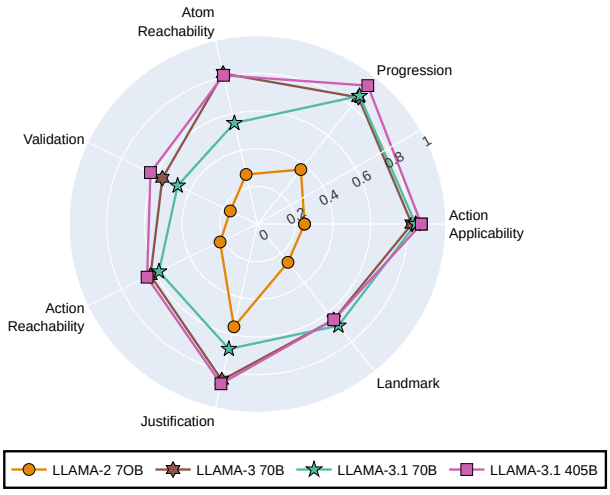



Figure 6: LLAMA versions comparison on ACPBench tasks.

showing improved plan generation of Granite model finetuned with QLoRA (Dettmers et al. 2023) on ACPBench tasks for 8 training domains.

**Performance over Time** One of our major motivations to generate and release the ACPBench collection is to encourage researchers to address the poor performance on these tasks and build models that are capable to perform reasoning required for better planning. We believe that without such benchmarks, the progress toward this goal is ad-hoc. We verify our belief by comparing the LLAMA model family's performance on ACPBench tasks over the past 6 months to see if there's been an improvement. Figure 6 presents the performance of LLAMA-2 70B, LLAMA-3 70B and

| Task | o1-preview | | o1-mini | |
|---|---|---|---|---|
| | Bool | MCQ | Bool | MCQ |
| Applicability | 93.08 | 95.38 | 90.77 | 76.92 |
| Progression | 97.69 | 96.15 | 91.54 | 88.46 |
| Reachability | 86.92 | 86.15 | 84.62 | 68.46 |
| Validation | 90.00 | 63.08 | 81.54 | 56.15 |
| Action Reach. | 72.50 | 85.00 | 55.83 | 70.00 |
| Justification | 88.46 | 89.23 | 80.00 | 83.85 |
| Landmark | 98.46 | 96.15 | 91.54 | 83.08 |
| Mean | 89.59 | 87.31 | 82.26 | 75.27 |

Table 5: o1 Reasoning Model performance on ACPBench.

LLAMA-3.1 70B and 405B on ACPBench tasks. We see that there is a significant jump in performance between LLAMA-2 and LLAMA-3. However, the difference in performance of LLAMA-3 70B and LLAMA-3.1 405B is not significant. This highlights the need for benchmarks that systematically captures the reasoning ability required for planning.

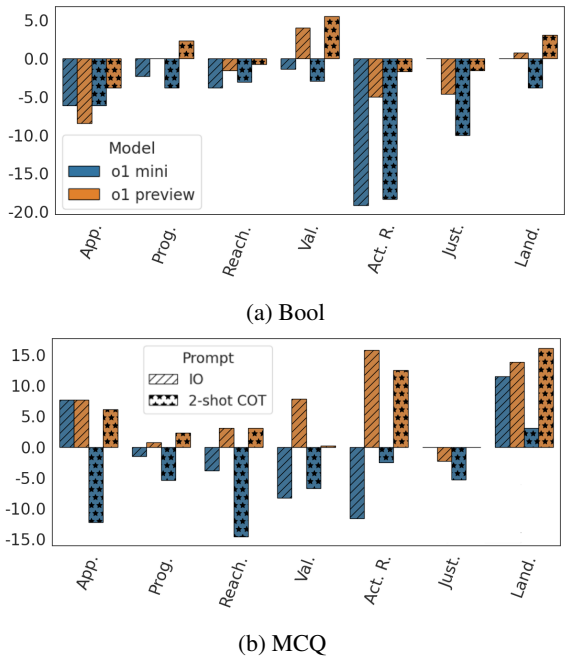
### 4.4 Reasoning Model: OpenAI o1

Recently, OpenAI released a series of LLM-based reasoning models called OpenAI o1 (OpenAI 2024b), that show significant improvement over GPT-4o on benchmarks that require reasoning. Although OpenAI o1 preview and mini are made available via similar APIs as previous LLMs, they do not truly fit the LLM category; rather, they are a system (or an agent) that makes multiple calls to LLMs before providing an answer. They are hence referred as Large Reasoning Models (Valmeekam, Stechly, and Kambhampati 2024). Table 5 presents the performance of OpenAI o1 on ACPBench tasks. While we acknowledge the difference between LLMs and Reasoning Models, we are interested in understanding the advantage provided by the reasoning over the the LLMs. To that end, Figure 7 shows the performance difference of OpenAI o1 models from the best performing LLMs. Our results indicate that OpenAI o1 models fail to yield performance gains for boolean questions, but demonstrate notable improvements on MCQs. Specifically, OpenAI o1 preview consistently performs better or equal to the best performing model for MCQs. The responses for MCQ tasks suggests that OpenAI o1 models consider each option individually, perform a case-by-case analysis, and only then select an option.

We would like to reiterate that while Figure 7 compares performance of OpenAI o1 with LLMs, the comparison is not even-handed due to below mentioned reasons:

- All our LLM experiments had a generated token limit of 1024; OpenAI o1 models did not have that limit. On average the number of tokens generated by OpenAI o1 preview for MCQ tasks, where we see the maximum improvement, was 5705 (this includes the completion token (3164) and the reasoning token (2542)).
- LLM evaluations are based on a single generation. We did not evaluate multi-turn prompts (such as self-consistency or self-reflection). OpenAI o1 models seem to internally make multiple calls to an LLM.

The OpenAI o1 evaluation is approx. 20 times more expensive than of GPT-4o. It remains to be seen if a multi-turn prompting of an open-sourced LLM like LLAMA-3.1 can achieve similar improvement with lower cost.




(a) Bool

(b) MCQ

Figure 7: Comparing OpenAI o1 models with the best LLM. Positive difference shows OpenAI o1 model performing better than the best of the LLMs. Negative difference is when OpenAI o1 model lags behind the best LLM.

## 5 Discussion and Future Work

In this work, we introduce ACPBench—a collection of datasets to evaluate the ability of LLMs to reason about action, change and planning. By evaluating 21 state-of-the-art LLMs of varying size, we find these models underperform, even the largest ones, especially on tasks such as plan validation and action reachability. On the other hand, we show that finetuning a small language model, Granite 8B, can improve its reasoning ability to bring it on par with the best performing models. Further, we observe that the fine-tuned model exhibits remarkable generalization ability to unseen domains in ACPBench as well as to a different task in PlanBench. Further, our investigation with OpenAI o1 reasoning model indicates that OpenAI's multi-turn approach yields improvements for multi-choice questions but fails to make an impact on boolean questions in ACPBench.

Performance of LLMs is known to be sensitive to prompt text as well as prompt style. Hence, it is possible to elicit better performance from each of these models with prompt engineering. In our work we do not modify prompts across models – our objective in the evaluation is to set a baseline. We hope our benchmark serves as a useful resource for improving LLM abilities. We encourage creative solutions (not limited to prompt engineering) to improve LLM performance across various tasks of ACPBench.

## Acknowledgments

Authors acknowledge the help of Maxwell Crouse, Asim Munawar, Ramón Fernandez Astudillo, Ibrahim Abdelaziz, YoungSuk Lee, and Pavan Kapanipathi at IBM Research.

## References

Abdin, M. I.; Jacobs, S. A.; Awan, A. A.; et al. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *CoRR*, abs/2404.14219.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *AIJ*, 69(1–2): 165–204.

Chu, Z.; Chen, J.; Chen, Q.; Yu, W.; He, T.; Wang, H.; Peng, W.; Liu, M.; Qin, B.; and Liu, T. 2024. Navigate through Enigmatic Labyrinth A Survey of Chain of Thought Reasoning: Advances, Frontiers and Future. In *ACL*. Association for Computational Linguistics.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Fink, E.; and Yang, Q. 1992. Formalizing Plan Justifications. In *Proc. CSCSI 1992*.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proc. AAAI 2020*, 9835–9842.

Fišer, D.; and Komenda, A. 2018. Fact-Alternating Mutex Groups for Classical Planning. *JAIR*, 61: 475–521.

Gou, Z.; Shao, Z.; Gong, Y.; Shen, Y.; Yang, Y.; Duan, N.; and Chen, W. 2024. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *ICLR*. OpenReview.net.

Granite Team, I. 2024. Granite 3.0 Language Models.

Handa, D.; Dolin, P.; Kumbhar, S.; Baral, C.; and Son, T. C. 2024. ActionReasoningBench: Reasoning about Actions with and without Ramification Constraints. *CoRR*, abs/2406.04046.

He, W.; Huang, C.; Xiao, Z.; and Liu, Y. 2023. Exploring the Capacity of Pretrained Language Models for Reasoning about Actions and Change. In *ACL*. Association for Computational Linguistics.

Hoffmann, J.; and Nebel, B. 2001. RIFO Revisited: Detecting Relaxed Irrelevance. In *Proc. ECP 2001*, 127–135.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *JAIR*, 22: 215–278.

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

Katz, M.; and Lee, J. 2023. K* Search Over Orbit Space for Top-k Planning. In *Proc. IJCAI 2023*.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In *Proc. AAAI 2020*, 9892–9899.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *Proc. ECAI 2010*, 335–340.

Lin, F. 2004. Discovering State Invariants. In *Proc. KR 2004*, 536–544.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; Zhang, S.; Deng, X.; Zeng, A.; Du, Z.; Zhang, C.; Shen, S.; Zhang, T.; Su, Y.; Sun, H.; Huang, M.; Dong, Y.; and Tang, J. 2024. AgentBench: Evaluating LLMs as Agents. In *ICLR*. OpenReview.net.

Ma, C.; Zhang, J.; Zhu, Z.; Yang, C.; Yang, Y.; Jin, Y.; Lan, Z.; Kong, L.; and He, J. 2024. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. *CoRR*, abs/2401.13178.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *NeurIPS*.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *AI Magazine*, 21(2): 35–55.

Mishra, M.; Stallone, M.; Zhang, G.; Shen, Y.; Prasad, A.; et al. 2024. Granite Code Models: A Family of Open Foundation Models for Code Intelligence. *CoRR*, abs/2405.04324.

MistralAI. 2024. Mixtral 8x22B. https://mistral.ai/news/mixtral-8x22b/.

OpenAI. 2024a. GPT 4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2024b. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering, and Usage of Landmarks in Planning. In *Proc. ECP 2001*, 174–182.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; Zhao, S.; Hong, L.; Tian, R.; Xie, R.; Zhou, J.; Gerstein, M.; Li, D.; Liu, Z.; and Sun, M. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *ICLR*. OpenReview.net.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proc. AAAI 2008*, 975–982.

Salerno, M.; Fuentetaja, R.; and Seipp, J. 2023. Eliminating Redundant Actions from Plans using Classical Planning. In *Proc. KR 2023*, 774–778.

Seipp, J.; Torralba, Á.; and Hoffmann, J. 2022. PDDL Generators. https://doi.org/10.5281/zenodo.6382173.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proc. NeurIPS 2023*.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *CVPR*, 10737–10746. Computer Vision Foundation / IEEE.

Shridhar, M.; Yuan, X.; Côté, M.-A.; Bisk, Y.; Trischler, A.; and Hausknecht, M. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Stein, K.; Fišer, D.; Hoffmann, J.; and Koller, A. 2024. AutoPlanBench: Automatically generating benchmarks for LLM planners from PDDL. arXiv:2311.09830 [cs.AI].

Suzgun, M.; Scales, N.; Schärli, N.; et al. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *ACL (Findings)*, 13003–13051. Association for Computational Linguistics.

Valmeekam, K.; Marquez, M.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023a. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In *Proc. NeurIPS 2023*, 38975–38987.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023b. On the Planning Abilities of Large Language Models - A Critical Investigation. In *Proc. NeurIPS 2023*.

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

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2024a. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6): 186345.

Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R. K.; and Lim, E. 2023. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In *ACL*. Association for Computational Linguistics.

Wang, X.; Wang, Z.; Liu, J.; Chen, Y.; Yuan, L.; Peng, H.; and Ji, H. 2024b. MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback. In *ICLR*. OpenReview.net.

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 *Proc. NeurIPS 2022*, 24824–24837.

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. In *ICML*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Proc. NeurIPS 2023*.

Zhu, L.; and Givan, R. 2003. Landmark Extraction via Planning Graph Propagation. In *ICAPS 2003 Doctoral Consortium*, 156–160.