

# ACTIONREASONINGBENCH: REASONING ABOUT ACTIONS WITH AND WITHOUT RAMIFICATION CONSTRAINTS

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Reasoning about Actions and Change (RAC) has historically played a pivotal role in solving foundational AI problems, such as the frame problem. It has driven advancements in AI fields, such as non-monotonic and commonsense reasoning. RAC remains crucial for AI systems that operate in dynamic environments, engage in interactive scenarios, or rely on commonsense reasoning. Despite substantial advances made by Large Language Models (LLMs) in various AI domains, their performance in RAC remains underexplored. To address this gap, we introduce a new diagnostic benchmark, ACTIONREASONINGBENCH, which encompasses 8 domains and includes questions for up to 19 action sequences. This benchmark rigorously evaluates LLMs across six key RAC dimensions: *Fluent Tracking*, *State Tracking*, *Action Executability*, *Effects of Actions*, *Numerical RAC*, and *Composite Questions*. LLMs demonstrate average accuracy rates of 73.55%, 65.63%, 58.73%, and 62.38% on the former four dimensions, which are frequently discussed in RAC literature. However, the performance on the latter two dimensions, which introduce complex and novel reasoning questions, the average performance of LLMs is lowered to 33.16% and 51.19%, respectively, reflecting a 17.9% performance decline. We also introduce new ramification constraints to capture the indirect effects of actions, providing deeper insights into RAC challenges. Our evaluation of state-of-the-art LLMs, including both open-source and commercial models, reveals challenges across all RAC dimensions, particularly in handling ramifications, with GPT-4o failing to solve any question and o1-preview achieving a score of only 18.4%.

## 1 INTRODUCTION

Reasoning about actions and change (RAC) is a fundamental problem in artificial intelligence, with its roots tracing back to early work from the 1960s (McCarthy et al., 1963). Initially, research focused on developing logical systems capable of effectively modeling and reasoning about actions and their effects in a dynamic environment. One of the significant challenges in this domain has been succinctly expressing how actions influence changeable properties of the world, known as **fluents**. For example, consider the statement: “Moving an object from location X to location Y results in the object being at location Y.” While it is relatively straightforward to describe the direct effects on the affected fluents, such as the object’s location, it is much more complex to account for the unaffected fluents, a challenge known as the *frame problem*. This challenge becomes exacerbated when the descriptions involve relationships between fluents in a state, such as “an object can not be at two different places at the same time”. While such constraints simplify action descriptions by decoupling them from fluents, they introduce indirect effects, or **ramifications**. For example, the statement “A block is said to be clear if there isn’t any block on top of it” describes a ramification fluent, “clear” dependent on another fluent “on top of.”

It took multiple decades of research to create a comprehensive logical formalization that adequately addressed these issues. It involved the laborious creation of numerous handcrafted rules and logic detailing the effects and preconditions of actions (Reiter, 2001). However, these tools are limited since they rely on manual effort to translate natural language descriptions of actions and their effects into formal logic representations. To address this challenge, recent research in natural language

processing (NLP) has begun exploring the capabilities of LLMs in RAC tasks, as demonstrated by works of He et al. (2023), Spiliopoulou et al. (2022), and Banerjee et al. (2020). However, these studies have not systematically decomposed the complex RAC problem into multiple categories and overlook the critical ramifications of actions seen in real-world scenarios. To address this gap, we introduce ACTIONREASONINGBENCH, a diagnostic RAC benchmark that aims to pinpoint where modern state-of-the-art LLMs struggle.

We decompose the RAC task into six distinct categories—*Fluent Tracking*, *State Tracking*, *Action Executability*, *Effects of Actions*, *Numerical RAC*, and *Composite Questions*. The first four categories focus on assessing fundamental aspects of RAC, while the latter two introduce more complex and novel question types. The questions in every category span action sequences ranging from 1 to 19 steps, allowing us to test the RAC capabilities at a series of action sequence ranges. Additionally, we introduce ramification constraints to represent the indirect effect of actions. These constraints simplify action descriptions and align more closely with real-world conditions but introduce additional complexity, as highlighted by McIlraith (2000). Specifically, we expand the domains by adding ramification fluents with dependencies up to four levels deep, where actions propagate their effects through multiple layers.

Highlights of our benchmark, ACTIONREASONINGBENCH, along with the comparison to previous benchmarks on RAC, are presented in Table 1. We evaluate four LLMs—two open-source models, Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct (Dubey et al., 2024), as well as two leading proprietary models, GPT-4o (Achiam et al., 2023) and o1-preview (OpenAI, 2024). These LLMs were tested on ACTIONREASONINGBENCH across various RAC categories under different prompt settings, including Zero-shot-CoT (Kojima et al., 2022) and Few-shot-3 (Brown, 2020), to assess how performance varies based on these configurations.

Our findings indicate that LLMs face substantial challenges, particularly when addressing complex RAC questions. The average performance of all LLMs on the complex categories decreases by 17.88% compared to the first four basic categories. The best performing LLM, GPT-4o, achieves an average accuracy of 59.91% on these categories. Notably, GPT-4o failed to produce any correct answers for questions involving ramifications constraints, while the o1-preview model achieved an accuracy of only 18.42%. Performance was especially poor in categories like *Action Executability*, *Numerical RAC*, and *Composite Questions*, with further declines observed as the length of action sequences increased. Additionally, LLMs struggled with reasoning in scenarios that combined both true and false fluents, experiencing an average performance drop of 12.16% compared to tasks involving exclusively true or false fluents.

	PlanBench	TRAC	(Ours)
Number of domains	2	1	8
Number of queries	26k	15k	152k
Max Action Sequence length	48	3	19
Max number of objects	24	5	28
Binary Questions (T/F)	×	✓	✓
Free Answers	✓	×	✓
State Tracking	✓	✓	✓
Action Executability	✓	✓	✓
Fluent Tracking	×	×	✓
Effects of Actions	×	×	✓
Numerical Reasoning	×	×	✓
Composite Questions	×	×	✓
Ramifications Constraints	×	×	✓
Subcategories of Fluents	×	×	✓

Table 1: Differences between ACTIONREASONINGBENCH (Ours) and previous benchmarks on RAC. PlanBench (Valmeekam et al., 2024) ; TRAC (He et al., 2023)

## 2 RELATED WORKS

**Benchmarking reasoning capabilities of LLMs** Evaluating the reasoning capabilities of LLMs using synthetic datasets has become a key focus in NLP, with increasing efforts to create challenging benchmarks. Notable areas of interest include the legal reasoning (Fei et al., 2023; Guha et al., 2023), logical reasoning (Luo et al., 2024; Han et al., 2024; Patel et al., 2024; Parmar et al., 2024) arithmetic reasoning (Cobbe et al., 2021; Miao et al., 2021), temporal reasoning (Uddin et al., 2024; Fatemi et al., 2024), and commonsense reasoning (Onoe et al., 2021; Lin et al., 2021; Geva et al., 2021; Lourie et al., 2021). Despite this progress, RAC remains significantly under-explored, even though

108 it plays a crucial role in several of these reasoning tasks, such as commonsense and legal reasoning.  
109 To fill this gap, we create ACTIONREASONINGBENCH using synthetically generated data.  
110

111 **Evaluating RAC and Planning** Benchmarking planning capabilities of LLMs is a well-studied  
112 area, with recent works (Zheng et al., 2024; Xie et al., 2024) demonstrating the challenges LLMs  
113 face. While planning is a non-polynomial problem that remains inherently difficult to solve, we  
114 believe that RAC, which is a polynomial problem, is a prerequisite for effective planning. Without  
115 comprehending the effects of actions, LLMs are unlikely to construct coherent plans. In this work,  
116 we address this gap by proposing ACTIONREASONINGBENCH, a diagnostic benchmark for RAC.

117 Previous research, such as Banerjee et al. (2020), has investigated RAC capabilities in models  
118 like RoBERTa, focusing primarily on binary questions or single-word answers. Extending this,  
119 He et al. (2023) assessed LLMs on a broader range of question categories. However, only two  
120 of these—Action Executability and State Tracking—pertain directly to RAC, with the remainder  
121 addressing the broader domain of planning. Similarly, Valmeekam et al. (2024) introduced a bench-  
122 mark evaluating both RAC and planning, with a detailed emphasis on planning tasks but maintaining  
123 a limited focus on RAC categories, specifically State Tracking and Action Executability. A compar-  
124 ative analysis of the benchmarks from He et al. (2023), Valmeekam et al. (2024), and our proposed  
125 benchmark is summarized in Table 1.

126 In this work, we concentrate on RAC and develop a benchmark that encompasses a wider range of  
127 categories, enabling more precise identification of areas where LLMs underperform. Furthermore,  
128 our benchmark introduces novel constraints involving ramifications, incorporating the indirect ef-  
129 fects of actions. To the best of our knowledge, ramification constraints have not been addressed in  
130 any existing benchmark, marking a significant advancement in the evaluation of RAC capabilities.  
131

### 132 3 ACTIONREASONINGBENCH

133 This section provides a detailed overview of our benchmark, including its categorization, creation  
134 methodology, and validation process. A sample instance is presented in Appendix B, where we also  
135 describe the objects, actions, and fluents within the domain.  
136  
137

#### 138 3.1 QUESTION CATEGORIES

139 In our benchmark, the questions are organized into six distinct categories, each designed to assess a  
140 specific dimension of RAC. Below, we provide a detailed description of each category.  
141  
142

- 143 1. **Fluent Tracking** - Given the initial state and the sequence of actions performed, this cate-  
144 gory contains questions about the fluents, i.e. properties of the domain, of an object from  
145 the changed state. For instance, in the *Grippers* domain, a fluent-tracking question might  
146 be “*List all valid properties associated with ball2.*”
- 147 2. **State Tracking** - This category extends the concept of *Fluent Tracking*. It involves query-  
148 ing about the complete set of fluents in the final state. For instance, in the *Blocksworld*  
149 domain, a state-tracking question might be “*What are all the valid properties in this state?*”
- 150 3. **Action Executability** - This category encompasses two types of questions related to exe-  
151 cutability of actions. The types of questions within this category are as follows:  
152
  - 153 (a) Given an initial state, and a sequence of actions, the question focuses on identifying  
154 the first action in the sequence that is not executable.
  - 155 (b) Given an initial state and a sequence of actions leading to a final state, the task is to  
156 identify all actions that can be executed in the final state. For instance, in the *Visitall*  
157 domain, an action-executability question might be “*List all executable actions present*  
158 *in the current state.*”
- 159 4. **Effects of Actions** - This category contains questions that explore the outcomes of perform-  
160 ing a specific action. For instance, in the *Mystery* domain, an Effects-of-action question can  
161 be “*From the current state, the vehicle v0 moves from location l1 to l0, and has fuel-level*  
*f6 and f5, which properties of the state will be true now?*”

- 162 5. **Numerical RAC** - Questions requiring a numerical response fall under this category. These  
 163 questions may derive from any of the four previously mentioned categories. For example,  
 164 in the *Spanner* domain, a Numerical-RAC question can be “*What are the number of exe-*  
 165 *cutable actions in the current state?*”
- 166 6. **Composite Question** - This category contains questions that integrate multiple above-  
 167 mentioned categories, combining up to three distinct categories. These questions require  
 168 multiple steps of reasoning to arrive at the correct solution. For example, in the *Satellite*  
 169 domain, a composite question may combine aspects of *Fluent Tracking* and *Action Exe-*  
 170 *cutability*. An example of such a question could be “*List all the properties of the state for*  
 171 *satellite0 before the first infeasible action in the sequence?*”

### 172 3.2 FLUENT CATEGORIES

173 We further divide the fluents of all 8 domains into three distinct categories, each representing a  
 174 different aspect of ramifications within RAC.

- 175 1. **Static Properties** - These properties remain unchanged regardless of any action performed.  
 176 For instance, the property “*connected*” in the *Visitall* domain represents whether two loca-  
 177 tions are connected, a relationship that remains constant irrespective of the robot’s action,  
 178 which may involve moving, picking up, or placing down objects. In this domain, the con-  
 179 nectivity between locations remains unchanged, irrespective of any action.
- 180 2. **Base Fluents** - These fluents can change as a direct result of an action and do not depend on  
 181 other fluents. For example, in the *Grippers* domain, the fluent “*carry*” indicates whether  
 182 an object is being carried by a robot’s gripper. This fluent can change if the action pick or  
 183 drop is performed.
- 184 3. **Ramification Fluents** - These fluents are influenced indirectly by other fluents, and action  
 185 descriptions do not explicitly dictate their changes. Instead, they are determined by the  
 186 dependencies and relationships within the system. Ramification fluents are further divided  
 187 into two sub-categories:
- 188 (a) **Derived Fluents** - These fluents rely on the state of other fluents, reflecting a level of  
 189 dependency. Changes to them occur as a consequence of changes in the fluents they  
 190 depend on rather than through direct action. For instance, the fluent “*stable*” in the  
 191 *Blocksworld* domain is considered a derived fluent as its state depends on the fluents  
 192 “*clear*” and “*on\_table*”. This relationship can be described as: “*Blocks are stable*  
 193 *when clear and on the table*”. Furthermore, fluent “*clear*” is itself a derived fluent,  
 194 dependent on the fluent “*on*”, which makes “*stable*” a second-level indirect effect.
- 195 (b) **Self-Derived Fluents** - These fluents rely on constraints related to themselves rather  
 196 than other fluents. For example, in the *Depot* domain, the fluent “*at*” represents the  
 197 location of a truck, which can only be at one location at any given time. If the truck  
 198 is *at* location l0, it cannot simultaneously be *at* location l1. Such constraints are ex-  
 199 plicitly included in the domain description, for example, “*A truck can only be in one*  
 200 *location at a time*”.

201 Classification of every fluent across all 8 domains can be found in Appendix I. Furthermore, for each  
 202 fluent type, we generate questions involving negative fluents, i.e. fluents that are false, which allows  
 203 us to evaluate LLM’s comprehension of negation within RAC contexts.

### 204 3.3 DATASET STRUCTURE AND VARIATIONS

205 **Selected Domains** ACTIONREASONINGBENCH requires domains that facilitate the evaluation of  
 206 LLMs on both short and long sequences of meaningful actions, where the effects and preconditions  
 207 of these actions are succinctly described. Additionally, these domains should reflect real-world  
 208 scenarios. To meet these criteria, we selected 8 domains *Blocksworld*, *Depots*, *Driverlog*, *Grippers*,  
 209 *Mystery*, *Satellite*, *Spanner*, and *Visitall*—sourced from the International Planning Competition (IPC),  
 210 covering the years 1998 to 2014. These domains are commonly used as benchmarks for evaluating  
 211 advanced planning systems and provide a robust foundation for research in automated planning.  
 212 Appendix G provides a detailed description of each domain. Notably, even state-of-the-art LLMs  
 213

like GPT-4o are not capable of generating diverse domains or action sequences that conform to the precise constraints outlined in these domain descriptions, justifying the reliance on IPC domains.

**Domain Descriptions and Ramifications** The domains provided by the IPC are described using the Planning Domain Definition Language (PDDL), a formal language designed to model deterministic actions and state transitions for planning problems. Further information on PDDL can be found in Appendix H.1. In this study, we initially translated the PDDL-based domains into natural language. Subsequently, ramification constraints were introduced into these natural language descriptions. The process underwent validation by two experts in the RAC domain to ensure correctness. Given that the category *Action Executability* focuses on determining whether the action can be performed rather than analyzing its effect, we concentrated on the categories *Fluent Tracking*, *State Tracking*, and *Effects of Actions* when introducing ramifications.

**Action-Sequence Lengths** In order to fine-tune LLMs, we generate a comprehensive set of questions that span various action-sequence lengths, specifically 1, 5, 10, 15, and 19. This range is chosen to capture increasing complexities in RAC. We curate a distinct subset of questions with action-sequence lengths of 1, 10, and 19 for evaluation. This subset is selected to assess the model’s performance across the action sequences range.

**Answer Types** We formulate two distinct types of questions based on the expected answer format. The first type consists of binary questions, where the response is either True or False. The second type involves subjective answers, which encompass a range of multiple objects, actions, or fluents.

### 3.4 DATA CREATION & VALIDATION

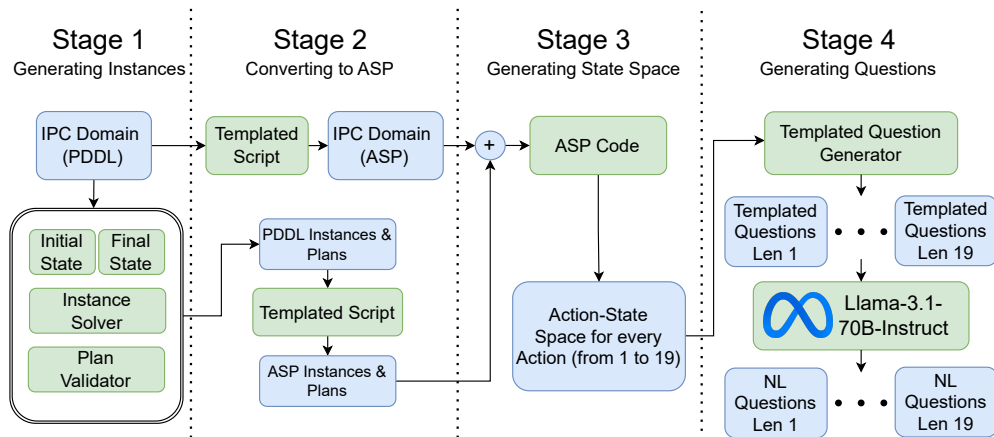


Figure 1: Overview of the question generation pipeline for ACTIONREASONINGBENCH. Blue blocks represent “Generated Data”, and green blocks represent “Code used in the pipeline”. Stage 1 involves generating states and plans using Helmert (2006) and validating them with Howey et al. (2004). In Stage 2, PDDL instances and plans are converted to ASP. Stage 3 computes the action-state space through ASP. Stage 4 generates questions using templates, which are then rephrased to natural language via Llama-3.1-70B-Instruct.

The question generation process follows a four-stage pipeline, as illustrated in Figure 1. The selected domains from the IPC are represented in PDDL (see Appendix G for examples). First, these PDDL representations are used to generate 10 pairs of initial and goal conditions. A PDDL solver (Helmert, 2006) and validator (Howey et al., 2004) are then employed to obtain and validate the action sequences necessary to transition from the initial to the goal state. In the second stage, the PDDL domains, instances, and action sequences are converted into Answer Set Programming (ASP) descriptions using Python-based templates.

In the third stage, ASP solvers are used to generate the action-state space and extract fluents for each state, along with identifying all executable and inexecutable actions. Further details on these

formal languages are provided in Appendix H. Finally, the fourth stage involves converting the action-state data into questions using a Python template. Up to three natural language variations are created for every object, action, and fluent. These templated sequences are then paraphrased to Llama-3.1-70B-Instruct to ensure they sound natural and avoid repetition in long action sequences. Three independent annotators review both the templated and the paraphrased versions to assess their naturalness, as detailed in Appendix D. Additionally, all eight domain descriptions were manually translated from PDDL to natural language.

### 3.5 DATA SPLITS

The benchmark was divided into two parts: one for training and the other for testing the LLMs, ensuring a balanced representation of question categories across the 8 domains. The *Composite Questions* category is slightly larger in the test set, as it combines multiple categories, leading to increased questions. Table 2 provides an overview of the distribution of questions and their corresponding categories across both the training and testing sets. The test set contains 3,498 questions, including 2,195 binary and 1,303 free-answer questions. Finally, we designed both zero-shot-CoT and few-shot-3 prompts for all the questions in the test set.

	Test Set	Train Set
Fluent Tracking	438	57,906
State Tracking	382	12,636
Action Executability	450	9,562
Effects of Actions	417	8,939
Numerical Reasoning	414	31,506
Composite Questions	1,397	28,688
Static Properties	237	12,458
Base Fluents	231	10,461
Derived Fluents	366	15,946
Self-Derived Fluents	390	23,436
Mixed Fluents	2,274	86,936
Total Unique Questions	3,498	149,237

Table 2: Overview of the test and train sets across Question and Fluent Categories. The “Mixed Fluents” category represents questions that involve more than one type of fluent.

## 4 EXPERIMENTS AND EVALUATION

**Models** To evaluate our benchmark, we tested four LLMs and employed two prompting techniques. The LLMs include two proprietary models—GPT-4o (Achiam et al., 2023) and o1-preview (OpenAI, 2024)—alongside two open-source models, Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct (Dubey et al., 2024). Each LLM was evaluated using both few-shot prompting with three examples (few-shot-3) Brown (2020) and zero-shot-CoT Kojima et al. (2022) prompting.

While the entire dataset requires reasoning abilities, the ramification subset involves the most complex and challenging reasoning tasks. Given that o1-preview is specifically optimized for reasoning tasks and incurs significantly higher costs compared to GPT-4o<sup>1</sup>, we restricted its use to the ramification subset, where its superior reasoning capabilities are expected to provide the greatest benefit. Utilizing o1-preview across the entire dataset would not be cost-effective, as its advantages would be less pronounced for simpler reasoning tasks.

To ensure a fair evaluation of the standalone reasoning capabilities of LLMs, we deliberately avoided incorporating external tools or systems. Although integrating formal solvers (e.g., PDDL-based planners) with LLMs could potentially improve performance, our primary focus is on assessing the intrinsic reasoning abilities of these models.

**Evaluation & Metrics** ACTIONREASONINGBENCH includes two types of answer formats, as outlined in Section 3.3: binary (true/false) and free-form responses. The evaluation process was performed separately for each answer type. For binary questions, we extracted “true” and “false” keywords from the model’s response and compared them to the ground truth. Since free-form answers can’t be evaluated using exact string matching, we employed human evaluation for the ramification questions. While human evaluation is highly accurate, it is not scalable, so we used Llama-3.1-70B-Instruct to assess all free-form responses. The specific prompt used to evaluate the LLMs and the correlation between Llama-3.1-70B-Instruct and human evaluations are detailed in Appendix F. For all experiments, we report the accuracy along with the standard error of the mean (SEM), calculated as  $SEM = \frac{\sigma}{\sqrt{n}}$ , where  $\sigma$  represents the standard deviation, and  $n$  is the sample size.

<sup>1</sup>As of Oct 2024, o1-preview is six times more expensive than GPT-4o <https://openai.com/api/pricing/>

**Fine-tuning** We fine-tuned the Llama-3.1-8B model using the training data split outlined in Section 3.5. Due to the limited computing power, we excluded any data samples that exceeded a context length of 4096 tokens. The fine-tuning process was performed separately for free-response and binary questions. Detailed information on the fine-tuning procedure is provided in Appendix E. All experiments were executed using 8xH100 GPUs.

## 5 RESULTS AND DISCUSSION

This section presents the results and analysis using ACTIONREASONINGBENCH. The Zero-shot-CoT results for each LLM on both the binary and free-response subsets of the test set are provided in Tables 5 and 3 respectively. Similarly, the Few-shot-3 results for each LLM evaluated on these same subsets are displayed in Tables 6 and 7. The detailed analysis of the effects when ramification constraints are incorporated into the descriptions is discussed in Section 5.1.

Action Seq.	Ques Categories	GPT-4o	Llama-8B-Inst	Llama-70B-Inst	Finetuned Llama-8B
1	Fluent Tracking	88.46 <sub>4.43</sub>	30.77 <sub>6.40</sub>	71.15 <sub>6.28</sub>	76.92 <sub>5.84</sub>
	State Tracking	73.33 <sub>6.59</sub>	28.89 <sub>6.76</sub>	64.44 <sub>7.14</sub>	75.56 <sub>6.41</sub>
	Action Executability	27.08 <sub>6.41</sub>	08.33 <sub>3.99</sub>	33.33 <sub>6.80</sub>	31.25 <sub>6.69</sub>
	Effects of Actions	82.50 <sub>6.01</sub>	20.00 <sub>6.32</sub>	67.50 <sub>7.41</sub>	60.53 <sub>7.93</sub>
	Numerical RAC	11.11 <sub>4.68</sub>	06.67 <sub>3.72</sub>	04.44 <sub>3.07</sub>	08.89 <sub>4.24</sub>
	Composite Questions	64.53 <sub>3.36</sub>	24.63 <sub>3.02</sub>	43.35 <sub>3.48</sub>	72.41 <sub>3.14</sub>
	Average	60.28 <sub>2.35</sub>	21.71 <sub>1.98</sub>	45.96 <sub>2.39</sub>	61.02 <sub>2.35</sub>
10	Fluent Tracking	82.00 <sub>5.43</sub>	36.00 <sub>6.79</sub>	62.00 <sub>6.86</sub>	80.00 <sub>5.66</sub>
	State Tracking	74.42 <sub>6.65</sub>	18.60 <sub>5.93</sub>	60.47 <sub>7.46</sub>	66.67 <sub>7.27</sub>
	Action Executability	34.09 <sub>7.15</sub>	11.36 <sub>4.78</sub>	40.91 <sub>7.41</sub>	47.73 <sub>7.53</sub>
	Effects of Actions	76.09 <sub>6.29</sub>	19.57 <sub>5.85</sub>	65.22 <sub>7.02</sub>	62.22 <sub>7.23</sub>
	Numerical RAC	10.20 <sub>4.32</sub>	02.04 <sub>2.02</sub>	06.12 <sub>3.42</sub>	10.20 <sub>4.32</sub>
	Composite Questions	59.11 <sub>3.45</sub>	16.26 <sub>2.59</sub>	45.32 <sub>3.49</sub>	58.62 <sub>3.46</sub>
	Average	57.01 <sub>2.37</sub>	17.01 <sub>1.80</sub>	45.98 <sub>2.39</sub>	55.66 <sub>2.39</sub>
19	Fluent Tracking	67.44 <sub>7.15</sub>	27.91 <sub>6.84</sub>	67.44 <sub>7.15</sub>	67.44 <sub>7.15</sub>
	State Tracking	75.51 <sub>6.14</sub>	16.33 <sub>5.28</sub>	51.02 <sub>7.14</sub>	65.31 <sub>6.80</sub>
	Action Executability	41.67 <sub>7.12</sub>	08.33 <sub>3.99</sub>	29.79 <sub>6.67</sub>	37.50 <sub>6.99</sub>
	Effects of Actions	76.60 <sub>6.18</sub>	14.89 <sub>5.19</sub>	46.81 <sub>7.28</sub>	61.70 <sub>7.09</sub>
	Numerical RAC	10.20 <sub>4.32</sub>	06.12 <sub>3.42</sub>	08.16 <sub>3.91</sub>	06.12 <sub>3.42</sub>
	Composite Questions	60.30 <sub>3.47</sub>	08.54 <sub>1.98</sub>	38.19 <sub>3.44</sub>	49.25 <sub>3.54</sub>
	Average	56.78 <sub>2.38</sub>	11.72 <sub>1.54</sub>	39.17 <sub>2.34</sub>	48.05 <sub>2.40</sub>

Table 3: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, and fine-tuned Llama-3.1-8B on the free-answer subset of the benchmark, evaluated without the ramifications constraints using the zero-shot-CoT. The results are categorized up by the action-sequence lengths and question categories.

**Performance across Domains** In our evaluation, GPT-4o demonstrated the highest performance on the *Grippers* domain and the lowest on the *Satellite* domain, with a performance gap of 15.53%. For both Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct, the best performance is also on the *Grippers* domain, but their worst performance occurs on the *Mystery* domain, with differences of 19.24% and 24.04%, respectively. Interestingly, the second lowest performing domain for GPT-4o is *Mystery*, while for the Llama models, it is *Satellite*. A detailed breakdown by domains is presented in Appendix C.3. This suggests that despite potential differences in pre-training data, these models exhibit a similar relative understanding of the domains.

Furthermore, when correlating these results with the State Space Complexity metric (described in App. G), traditionally used to gauge complexity for classical AI systems, we observe an inconsistent trend. This discrepancy implies that LLMs may employ different heuristics from those of traditional AI systems when tackling RAC problems, an observation that opens avenues for future exploration.

**Performance across Question Categories** LLMs perform well in *Fluent Tracking*, *State Tracking*, and *Effects of Actions*, demonstrating their strength in keeping track of changes. However, they

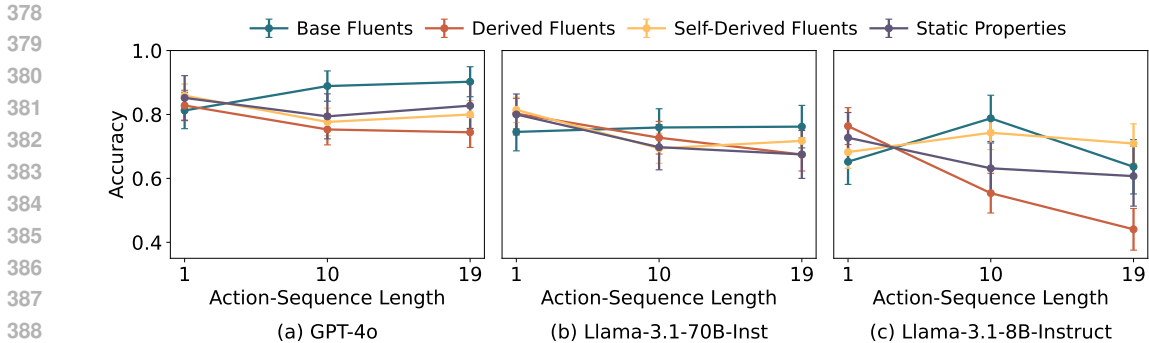


Figure 2: Performance for every Fluent Category for both binary and free-answer questions for every Action-Sequence length for GPT-4o, Llama-3.1-70B-Instruct, and Llama-3.1-8B-Instruct for Zero-shot-CoT prompt. Note: Bars represent SEM.

struggle with *Action Executability*, *Composite Questions*, and *Numerical RAC*. The performance of all LLMs on free-response complex questions highlights significant challenges, especially in the *Numerical RAC* category. This category reformulates existing question types into numerical formats, a domain where all tested LLMs exhibit notable difficulty. Specifically, performance on numerical questions related to *Action Executability* is 8.65% lower than on questions in the *Fluent Tracking* category. Previous research, such as (Ahn et al., 2024; McCoy et al., 2023), indicates that LLMs struggle with arithmetic reasoning and counting, which, when mixed with the RAC questions, likely contributes to the poor performance in the *Numerical RAC* category.

For *Composite Questions*, the combination of *Fluent Tracking* and *Action Executability* proves easier to answer than the combination of *State Tracking* and *Action Executability*, with a 16.32% performance difference. This can be attributed to the fact that the *State Tracking* category is a superset of the *Fluent Tracking* category, thereby explaining the observed difference in difficulty.

**Performance across Fluent Categories** As evident from Figure 2 and Table 9, across all LLMs examined in the study, a consistent trend emerges in which performance on *Static Properties* decreases as the length of action-sequence increases. While these static properties remain unchanged throughout the actions, they might get overlooked in longer sequences, likely due to their absence in the effect of any action. This phenomenon resembles the “needle in a haystack” challenge in long-context scenarios, where LLMs struggle to recall specific information embedded within a long context (Zhang et al., 2024). Conversely, *Base Fluents* maintain stable performance across all action sequences, indicating that the LLMs consistently capture the direct effects of actions. *Ramification fluents* exhibit a steady decline in performance as the sequence lengthens, particularly affecting the subcategory of *Derived Fluents*, which suggests that LLMs have more difficulty handling indirect effects. Finally, *Mixed Fluents*, which involve more than one fluent type, show a consistent decline in performance as the length of action sequences increases.

**Performance across Action-Sequence Lengths** Figure 3 illustrates the performance of the three models—GPT-4o, Llama-3.1-70B-Instruct, and Llama-3.1-8B-Instruct—across varying action sequence lengths in a zero-shot-CoT setting. The results combine both binary and free-response formats, with detailed data available in Tables 5 and 3. Generally, model accuracy declines as the action-sequence length increases, a pattern that holds for most categories. However, GPT-4o deviates from this trend in the *State Tracking* category, where performance first decreases and then improves. Since this trend is not observed with the other two models, and the results lie within the margin of error, we believe this is an outlier. In contrast, the *Effects of Actions* category consistently deviates from this trend, likely due to the nature of the task, which focuses on changes resulting from the last action, making it less dependent on the sequence of actions.

**Model Parameters and Fine-tuning** As demonstrated in Tables 5 and 3, the Llama-3.1-70B-Instruct model consistently outperforms the smaller Llama-3.1-8B-Instruct model, with an average performance improvement of 20.84%. This improvement is likely due to the larger model’s supe-



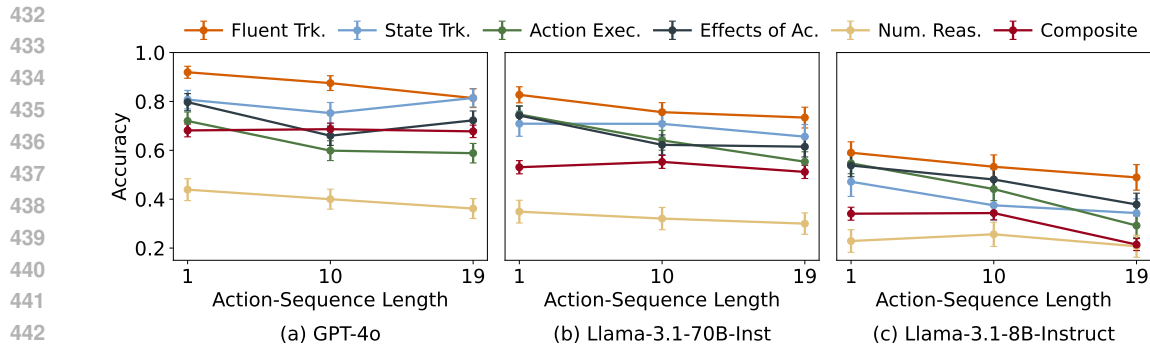


Figure 3: Performance for every Question Category for both binary and free-answer questions for every Action-Sequence length for GPT-4o, Llama-3.1-70B-Instruct, and Llama-3.1-8B-Instruct for Zero-shot-CoT prompt. Note: Bars represent SEM.

rior reasoning capabilities stemming from its increased number of parameters. A similar trend is observed when comparing GPT-4o to Llama-3.1-70B-Instruct, where GPT-4o exhibits an average performance increase of 8.64%. Although the specific size of GPT-4o remains undisclosed, it is widely speculated to be in the trillions of parameters<sup>2</sup>. Notably, fine-tuning the Llama-3.1-8B model on the training set resulted in substantial gains in both binary and free-answer tasks, with an average performance increase of 33.68% across the test set, even outperforming GPT-4o by 4.2%.

**Impact of Few-Shot Examples on Model Performance** As shown in Tables 5 and 6, the inclusion of few-shot examples for binary answer categories does not significantly enhance model accuracy. This limitation is especially pronounced in models such as GPT-4o and Llama-3.1-70B-Instruct, which exhibit a relative performance decline of approximately 3.5% compared to zero-shot-CoT conditions. We hypothesize that this decrease may be attributed to the few-shot examples inadvertently leading the model toward detecting spurious correlations. In contrast, the model relies more heavily on its internal reasoning capabilities in a zero-shot-CoT setting, potentially mitigating biases introduced by example-driven patterns. However, as seen from Tables 3 and 7, the few-shot approach shows effectiveness in free-answer questions only open-source LLMs.

**LLMs Struggle with Negative Fluents** Our study reveals a consistent pattern across all the LLMs examined as demonstrated in Table 10, wherein their performance declines when tasked with questions involving negative fluents compared to those focused on fluents that are true. Specifically, we observed a 12.16% decrease in accuracy on these negative fluent tasks. Furthermore, when questions required reasoning about both true and false fluents simultaneously, LLMs exhibited competence in identifying the true fluents but demonstrated difficulty in correctly recalling the false ones.

## 5.1 RAMIFICATIONS RESULTS

As discussed in Section 4, the performance on ramification fluents is evaluated for two LLMs: GPT-4o, the highest-performing LLM, and o1-preview, the most recent state-of-the-art LLM. Table 4 presents the performance of both models when ramification constraints are introduced. Further examples of the model responses to ramification-related questions can be found in the Appendix J.

**GPT-4o Performance** GPT-4o did not answer any ramification-related questions correctly, as depicted in Table 4. Upon manual inspection of its outputs, it became evident that GPT-4o frequently failed to mention ramification fluents, even when these were explicitly detailed in the domain description. In instances where it did address ramification fluents, the responses were incorrect or incomplete, with some fluents being omitted. We hypothesize that GPT-4o may have encountered the domain data during pre-training and relied on memorized effects of actions, as the experimental domains were derived from publicly available IPC datasets. Since the ramification fluents were

<sup>2</sup>GPT-4o and Gemini 1.5 Pro: How the New AI Models Compare - CNET

Action Sequence	Question Categories	Free Answer		Binary Questions	
		GPT-4o	o1-preview	GPT-4o	o1-preview
1	Fluent Tracking	00.00 <sub>00.00</sub>	00.00 <sub>00.00</sub>	71.43 <sub>17.10</sub>	100.00 <sub>00.00</sub>
	State Tracking	00.00 <sub>00.00</sub>	20.00 <sub>17.89</sub>	100.00 <sub>00.00</sub>	100.00 <sub>00.00</sub>
	Effects of Actions	00.00 <sub>00.00</sub>	40.00 <sub>21.91</sub>	71.43 <sub>17.10</sub>	57.14 <sub>18.70</sub>
	Average	00.00 <sub>00.00</sub>	25.00 <sub>12.49</sub>	80.95 <sub>08.57</sub>	85.71 <sub>07.60</sub>
10	Fluent Tracking	00.00 <sub>00.00</sub>	00.00 <sub>00.00</sub>	57.14 <sub>18.70</sub>	57.14 <sub>18.70</sub>
	State Tracking	00.00 <sub>00.00</sub>	33.33 <sub>19.24</sub>	42.86 <sub>18.70</sub>	-
	Effects of Actions	00.00 <sub>00.00</sub>	14.28 <sub>13.22</sub>	71.43 <sub>17.10</sub>	100.00 <sub>00.00</sub>
	Average	00.00 <sub>00.00</sub>	23.07 <sub>11.68</sub>	57.14 <sub>10.80</sub>	78.57 <sub>10.90</sub>
19	Fluent Tracking	00.00 <sub>00.00</sub>	33.33 <sub>27.21</sub>	42.86 <sub>18.70</sub>	57.14 <sub>18.70</sub>
	State Tracking	00.00 <sub>00.00</sub>	00.00 <sub>00.00</sub>	57.14 <sub>18.70</sub>	-
	Effects of Actions	00.00 <sub>00.00</sub>	00.00 <sub>00.00</sub>	71.43 <sub>17.10</sub>	85.71 <sub>13.20</sub>
	Average	00.00 <sub>00.00</sub>	07.69 <sub>07.38</sub>	57.14 <sub>10.80</sub>	71.43 <sub>12.10</sub>

Table 4: Performance comparison of GPT-4o and o1-preview on both the free-answer and binary question subset of the benchmark, evaluated with the ramifications constraints using the zero-shot-CoT. The results are categorized up by the action-sequence lengths and question categories with a “-” indicating no response.

manually created and integrated for this study, this evaluation assesses the model’s reasoning abilities without leveraging pre-existing knowledge. This likely explains why GPT-4o failed to generate ramification fluents, as its pre-training included only the original fluents from the domains.

**o1-preview Performance** o1-preview, a recently developed LLM optimized for reasoning tasks and incorporating a novel run-time inference mechanism OpenAI (2024), demonstrated significantly better performance on ramification-related questions compared to GPT-4o, as presented in Table 4. A detailed review of its outputs showed that o1-preview can correctly identify most ramification fluents. However, the model struggles with fluents involving negation, which consistently poses a challenge. It often omitted certain fluents in its final answers and, in some cases, incorrectly evaluated the ramification fluents.

## 6 CONCLUSION

In this work, we introduced a new diagnostic benchmark, ACTIONREASONINGBENCH, designed to comprehensively evaluate the performance of large language models (LLMs) on reasoning about actions and change (RAC). By assessing various LLMs across eight domains and six key dimensions of RAC, our findings indicate that while LLMs demonstrate moderate proficiency on traditional RAC tasks, such as *Fluent Tracking*, *State Tracking*, *Action Executability*, and *Effects of Actions*, they exhibit significant challenges when addressing more complex and novel questions, particularly in areas like *Numerical RAC* and *Composite Questions*, with an average performance drop of 17.9%. This highlights a substantial gap in the current ability of LLMs to handle complex reasoning tasks.

Additionally, we explored the indirect effects of actions, known as ramifications, where even state-of-the-art models show considerable limitations. For example, GPT-4o could not solve any questions involving ramifications, and the o1-preview model achieved a low score of 18.4%. These results underscore the pressing need for further research and advancements in RAC reasoning, especially in addressing indirect effects and more advanced reasoning tasks. Improving LLMs’ capabilities in these areas is crucial for enhancing their performance in dynamic and interactive environments. Our findings reinforce the challenges LLMs face in RAC tasks, which, despite their polynomial complexity in cases without ramifications, serve as prerequisites for more advanced planning problems that are often NP-complete or beyond. Improving LLM performance in RAC is a pivotal step toward overcoming broader planning challenges.

## REFERENCES

- 540  
541  
542 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-  
543 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical  
544 report. *arXiv preprint arXiv:2303.08774*, 2023.
- 545 Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models  
546 for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.
- 547 Pratyay Banerjee, Chitta Baral, Man Luo, Arindam Mitra, Kuntal Pal, Tran C Son, and Neeraj  
548 Varshney. Can transformers reason about effects of actions? *arXiv preprint arXiv:2012.09938*,  
549 2020.
- 550 Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- 551 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,  
552 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to  
553 solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>, 2021.
- 554 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha  
555 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.  
556 *arXiv preprint arXiv:2407.21783*, 2024.
- 557 Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch,  
558 Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating  
559 llms on temporal reasoning. *arXiv preprint arXiv:2406.09170*, 2024.
- 560 Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen,  
561 Zongwen Shen, and Jidong Ge. LawBench: Benchmarking Legal Knowledge of Large Language  
562 Models, September 2023.
- 563 Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving  
564 to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.
- 565 Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did Aristotle  
566 Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies, January  
567 2021.
- 568 Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Aditya K, Alex Chohlas-  
569 Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, Dmitry Talisman,  
570 Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory Dickinson, Haggai Porat, Jason  
571 Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan Choi, Kevin Tobia, Margaret  
572 Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt,  
573 Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom  
574 Zur, Varun Iyer, and Zehua Li. LegalBench: A Collaboratively Built Benchmark for Measur-  
575 ing Legal Reasoning in Large Language Models. *Advances in Neural Information Processing*  
576 *Systems*, 36:44123–44279, December 2023.
- 577 Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James  
578 Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo,  
579 Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, An-  
580 song Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscin-  
581 ski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying,  
582 Arman Cohan, and Dragomir Radev. FOLIO: Natural Language Reasoning with First-Order  
583 Logic, May 2024.
- 584 Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, Christian Muise, Ronald Brachman, Francesca  
585 Rossi, and Peter Stone. *An introduction to the planning domain definition language*, volume 13.  
586 Springer, 2019.
- 587 Weinan He, Canming Huang, Zhanhao Xiao, and Yongmei Liu. Exploring the capacity of pretrained  
588 language models for reasoning about actions and change. In *Proceedings of the 61st Annual*  
589 *Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4629–  
590 4643, 2023.

- 594 Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:  
595 191–246, 2006.
- 596
- 597 Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects  
598 and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with*  
599 *Artificial Intelligence*, pp. 294–301. IEEE, 2004.
- 600 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large  
601 language models are zero-shot reasoners. *Advances in neural information processing systems*,  
602 35:22199–22213, 2022.
- 603
- 604 Bill Yuchen Lin, Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Xiang Ren, and William W. Cohen.  
605 Differentiable Open-Ended Commonsense Reasoning, June 2021.
- 606 Do Xuan Long, Hai Nguyen Ngoc, Tiviatis Sim, Hieu Dao, Shafiq Joty, Kenji Kawaguchi, Nancy F  
607 Chen, and Min-Yen Kan. LLMs are biased towards output formats! systematically evaluating and  
608 mitigating output format bias of LLMs. *arXiv preprint arXiv:2408.08656*, 2024.
- 609
- 610 Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. UNICORN on RAINBOW:  
611 A Universal Commonsense Reasoning Model on a New Multitask Benchmark, March 2021.
- 612
- 613 Man Luo, Shrinidhi Kumbhar, Ming shen, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak  
614 Aditya, and Chitta Baral. Towards LogiGLUE: A Brief Survey and A Benchmark for Analyzing  
615 Logical Reasoning Capabilities of Language Models, March 2024.
- 616
- 617 John McCarthy et al. *Situations, actions, and causal laws*. Comtex Scientific, 1963.
- 618
- 619 R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. Embers  
620 of autoregression: Understanding large language models through the problem they are trained to  
621 solve. *arXiv preprint arXiv:2309.13638*, 2023.
- 622
- 623 Sheila A McIlraith. Integrating actions and state constraints: A closed-form solution to the ramifi-  
624 cation problem (sometimes). *Artificial Intelligence*, 116(1-2):87–121, 2000.
- 625
- 626 Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A Diverse Corpus for Evaluating and Devel-  
627 oping English Math Word Problem Solvers, June 2021.
- 628
- 629 Yasumasa Onoe, Michael JQ Zhang, Eunsol Choi, and Greg Durrett. Creak: A dataset for common-  
630 sense reasoning over entity knowledge. *arXiv preprint arXiv:2109.01653*, 2021.
- 631
- 632 OpenAI. Openai o1 system card, 2024. URL <https://cdn.openai.com/o1-system-card-20240917.pdf>. Accessed: 2024-09-30.
- 633
- 634 Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty,  
635 Arindam Mitra, and Chitta Baral. Towards Systematic Evaluation of Logical Reasoning Ability  
636 of Large Language Models, April 2024.
- 637
- 638 Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varsh-  
639 ney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of  
640 large language models. *arXiv preprint arXiv:2406.17169*, 2024.
- 641
- 642 Raymond Reiter. *Knowledge in action: logical foundations for specifying and implementing dy-*  
643 *namical systems*. 2001.
- 644
- 645 Evangelia Spiliopoulou, Artidoro Pagnoni, Yonatan Bisk, and Eduard Hovy. Events realm: Event  
646 reasoning of entity states via language models. In *Proceedings of the 2022 Conference on Empir-*  
647 *ical Methods in Natural Language Processing*, pp. 1982–1997, 2022.
- 648
- 649 Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen.  
650 Let me speak freely? a study on the impact of format restrictions on performance of large language  
651 models. *arXiv preprint arXiv:2408.02442*, 2024.
- 652
- 653 Md Nayem Uddin, Amir Saeidi, Divij Handa, Agastya Seth, Tran Cao Son, Eduardo Blanco,  
654 Steven R Corman, and Chitta Baral. Unseentimeqa: Time-sensitive question-answering beyond  
655 LLMs’ memorization. *arXiv preprint arXiv:2407.03525*, 2024.

648 Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kamb-  
649 hampati. Planbench: An extensible benchmark for evaluating large language models on planning  
650 and reasoning about change. *Advances in Neural Information Processing Systems*, 36, 2024.  
651

652 Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and  
653 Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint*  
654 *arXiv:2402.01622*, 2024.

655 Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluat-  
656 ing text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.  
657

658 Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu,  
659 and Zhangyang Wang. Found in the middle: How language models use long contexts better via  
660 plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*, 2024.

661 Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova,  
662 Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. Natural plan: Benchmarking llms on  
663 natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701

## A LIMITATIONS & FUTURE WORK

While ACTIONREASONINGBENCH provides a diagnostic assessment of LLMs on RAC, it represents an early step in this area and has several limitations, including but not limited to the following:

1. Although RAC is not inherently dependent on the English language, the current version of ACTIONREASONINGBENCH is limited to questions formulated in English.
2. Although there are more complex types of RAC, including incorporating more reasoning types, exploring those remains beyond the scope of this work and is left as a direction for future research.
3. While the IPC domains in our work cover many scenarios, they may introduce a bias towards planning-specific domains. Expanding the dataset to include more domains could help mitigate this bias.
4. Despite our efforts to evaluate a variety of LLMs, including open and proprietary LLMs, our assessment did not cover models with different architectures or training approaches due to resource limitations.
5. Recent studies by Long et al. (2024) and Tam et al. (2024) indicate that LLM performance can fluctuate depending on the prompt format. This variation may lead to a marginal improvement in the performance of LLMs on ACTIONREASONINGBENCH.
6. Our free-answer evaluation, which relies on prompting Llama-3.1-70B-Instruct, isn't perfect and reflects an ongoing challenge of evaluating the free-answers within the NLP community.

## B DESCRIBING AN INSTANCE FROM ACTIONREASONINGBENCH

Blocksworld domain with Ramifications for a single sequence of action for Fluent Tracking

[DOMAIN DESCRIPTION]

A block can only be picked up if it is clear, on the table, and the hand is empty, resulting in the block being held. A held block can be put down, placing it back on the table. Blocks can be stacked if the first block is held and the second block is clear, causing the first block to rest on top of the second. Unstacking occurs when the hand is empty, the first block is clear, and on top of the second, resulting in the first block being held again. A block can't be at two locations at the same time and is considered clear if nothing is on top of it and it's not held, and the hand is empty if it's not holding anything. Blocks are stable when clear and on the table, and they can be painted if stable and the hand is empty. A block is considered on display if it can be painted and has no other block on top of it.

[INITIAL CONDITIONS]

Block b1 is situated on the table, block b2 is not stacked with any other block, block b2 is also on the table, block b3 is not stacked with any other block, block b3 is positioned on top of block b7, block b4 is stacked on top of block b1, block b5 is not stacked with any other block, block b5 is placed on top of block b4, block b6 is on the table, block b7 is stacked on top of block b6, and the hand is empty.

[QUESTION]

Starting from the initial condition, the following actions are taken: block b3 is unstacked from the top of block b7 to achieve the current state. In this state, what are the valid properties (including both affirmative and negated properties) for b7? If there are no valid properties, write None.

In the domain description, actions and their corresponding effects on the state are outlined, including the necessary conditions for performing these actions. The initial conditions describe the starting configuration of objects within the domain. A typical scenario involves executing a sequence of actions that alter the configuration of the state, followed by a query. In the example provided, the

756 question falls under the category of fluent tracking, which asks about a specific set of properties  
757 associated with the object "block b7" after one action has been performed.

758  
759 The objects involved in this example are as follows:

- 760 • block b1
- 761 • block b2
- 762 • block b3
- 763 • block b4
- 764 • block b5
- 765 • block b6
- 766 • block b7
- 767 • block b7
- 768 • hand
- 769

770 The properties of the "block" object include:

- 771
- 772 • Block X on top of block Y
- 773 • Block X is on the table
- 774 • Block X is clear
- 775 • Block X is stable
- 776 • Block X can be painted
- 777 • Block X can be displayed
- 778 • Block X is held
- 779
- 780

781 The properties of the "hand" object are:

- 782
- 783 • The hand is empty
- 784 • The hand is holding block x
- 785

786 Several ramification constraints (i.e., properties that depend on other properties) are present in  
787 "Blocksworld":

- 788
- 789 • **Clear:** depends on the properties "on top of" and "held"
- 790 • **Stable:** depends on the properties "clear" and "on the table"
- 791 • **Paint:** depends on the properties "stable" and "hand is empty"
- 792 • **Display:** depends on the properties "painted" and "on top"
- 793 • **On top of:** depends on itself, since a block can't be at two locations at the same time
- 794 • **Hand is holding a block:** depends on itself, since the hand cannot hold two blocks at the  
795 same time
- 796

797 The valid actions that can be performed within this domain include:

- 798
- 799 • Picking up a block
- 800 • Putting down a block
- 801 • Stacking a block on top of another block
- 802 • Unstacking a block from top of another block
- 803

## 804 C ADDITIONAL RESULTS

### 805 C.1 ZERO-SHOT-CoT BINARY RESULT

806  
807 Table 5 shows the results on GPT-4o, Llama-3.1-8B-Instruct, and Llama-3.1-70B-Instruct on zero-  
808 shot-CoT prompting.  
809

Action Seq.	Ques Categories	GPT-4o	Llama-8B-Inst	Llama-70B-Inst	Fine-tuned Llama-8B
1	Fluent Tracking	94.44 <sub>2.70</sub>	80.00 <sub>4.78</sub>	90.12 <sub>3.31</sub>	89.06 <sub>3.90</sub>
	State Tracking	85.94 <sub>4.35</sub>	80.00 <sub>8.00</sub>	79.41 <sub>6.93</sub>	98.55 <sub>1.44</sub>
	Action Executability	93.14 <sub>2.50</sub>	79.12 <sub>4.26</sub>	94.12 <sub>2.33</sub>	94.74 <sub>3.62</sub>
	Effects of Actions	78.49 <sub>4.26</sub>	70.89 <sub>5.11</sub>	77.17 <sub>4.38</sub>	96.55 <sub>2.40</sub>
	Numerical RAC	62.82 <sub>5.47</sub>	42.11 <sub>8.01</sub>	57.38 <sub>6.33</sub>	49.37 <sub>5.62</sub>
	Composite Questions	74.56 <sub>4.08</sub>	50.43 <sub>4.62</sub>	67.39 <sub>3.99</sub>	94.42 <sub>1.57</sub>
	Average	81.26 <sub>1.71</sub>	66.43 <sub>2.30</sub>	77.76 <sub>1.85</sub>	87.76 <sub>1.43</sub>
10	Fluent Tracking	91.43 <sub>3.35</sub>	68.42 <sub>6.16</sub>	84.93 <sub>4.19</sub>	89.86 <sub>3.63</sub>
	State Tracking	75.86 <sub>5.62</sub>	76.19 <sub>9.29</sub>	86.21 <sub>6.40</sub>	96.43 <sub>2.48</sub>
	Action Executability	70.87 <sub>4.48</sub>	65.67 <sub>5.80</sub>	74.49 <sub>4.40</sub>	81.82 <sub>8.22</sub>
	Effects of Actions	61.22 <sub>4.92</sub>	63.86 <sub>5.27</sub>	60.82 <sub>4.96</sub>	96.67 <sub>2.32</sub>
	Numerical RAC	55.21 <sub>5.08</sub>	65.52 <sub>8.83</sub>	54.39 <sub>6.60</sub>	51.14 <sub>5.33</sub>
	Composite Questions	82.96 <sub>3.24</sub>	71.00 <sub>4.54</sub>	68.92 <sub>3.80</sub>	88.79 <sub>2.07</sub>
	Average	72.50 <sub>1.89</sub>	67.79 <sub>2.47</sub>	70.12 <sub>2.04</sub>	84.06 <sub>1.59</sub>
19	Fluent Tracking	91.53 <sub>3.63</sub>	67.35 <sub>6.70</sub>	77.27 <sub>5.16</sub>	85.71 <sub>4.68</sub>
	State Tracking	85.94 <sub>4.35</sub>	83.33 <sub>8.78</sub>	81.82 <sub>5.81</sub>	96.36 <sub>2.52</sub>
	Action Executability	66.67 <sub>4.60</sub>	53.66 <sub>7.79</sub>	66.99 <sub>4.63</sub>	77.27 <sub>6.32</sub>
	Effects of Actions	70.00 <sub>4.83</sub>	54.69 <sub>6.22</sub>	69.32 <sub>4.92</sub>	97.10 <sub>2.02</sub>
	Numerical RAC	50.00 <sub>5.21</sub>	42.42 <sub>8.60</sub>	47.54 <sub>6.39</sub>	66.29 <sub>5.01</sub>
	Composite Questions	78.42 <sub>3.49</sub>	53.09 <sub>5.54</sub>	68.97 <sub>3.84</sub>	86.18 <sub>2.20</sub>
	Average	72.31 <sub>1.91</sub>	56.64 <sub>2.93</sub>	68.24 <sub>2.07</sub>	84.62 <sub>1.53</sub>

Table 5: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, and fine-tuned Llama-3.1-8B on the binary subset (True/False) of the benchmark, evaluated without the ramifications constraints using the zero-shot-CoT. The results are categorized up by the action-sequence lengths and question categories.

## C.2 FEW-SHOT-3 RESULTS

Table 6 and 7 presents the results using the Few-shot-3 setting. These tables support the results showed in Section 5.

## C.3 RESULTS BY DOMAINS

Table 8 shows the results of the binary questions across every domain in our benchmark.

## C.4 RESULTS BY FLUENTS

Tables 9, 10 show the results of the binary questions across fluent types in our benchmark.

## D DATA VERIFICATION

To ensure the soundness of our benchmark, we employed three independent annotators who had no prior involvement with the project. Their task was to evaluate the naturalness of the sentences by assigning a score from 1 to 5, where 1 indicates the least natural and 5 most natural. To make sure that rephrasing the templated questions using Llama-3.1-70B-Instruct helps, we sample from each domain in the dataset was represented by 5 randomly sampled instances, resulting in a total of 65 samples across all domains for both templated questions and rephrased questions, resulting in a total of 130 samples. The annotators were provided with both the sampled instances and the following instructions:

Instruction to the Annotators

Rate the Prompts from 1 to 5, based on how natural they appear in English.

Table 11 summarizes the naturalness scores assigned by annotators across all domains in ACTION-REASONINGBENCH. The templated dataset received an average naturalness score of 4.2 out of



Action Seq.	Ques Categories	GPT-4o	Llama-8B-Inst	Llama-70B-Inst
1	Fluent Tracking	85.00 <sub>3.57</sub>	68.42 <sub>4.77</sub>	82.65 <sub>3.82</sub>
	State Tracking	91.57 <sub>3.05</sub>	71.95 <sub>4.96</sub>	89.29 <sub>3.37</sub>
	Action Executability	95.10 <sub>2.14</sub>	84.31 <sub>3.60</sub>	93.14 <sub>2.50</sub>
	Effects of Actions	74.19 <sub>4.54</sub>	73.91 <sub>4.58</sub>	69.89 <sub>4.76</sub>
	Numerical RAC	58.54 <sub>5.44</sub>	56.76 <sub>5.76</sub>	56.10 <sub>5.48</sub>
	Composite Questions	82.66 <sub>2.40</sub>	58.37 <sub>3.23</sub>	74.30 <sub>2.77</sub>
	Average	81.92 <sub>1.45</sub>	67.26 <sub>1.80</sub>	77.26 <sub>1.58</sub>
10	Fluent Tracking	84.54 <sub>3.67</sub>	69.66 <sub>4.87</sub>	82.47 <sub>3.86</sub>
	State Tracking	91.55 <sub>3.30</sub>	64.18 <sub>5.86</sub>	86.11 <sub>4.08</sub>
	Action Executability	73.79 <sub>4.33</sub>	63.00 <sub>4.83</sub>	74.76 <sub>4.28</sub>
	Effects of Actions	59.18 <sub>4.96</sub>	53.68 <sub>5.12</sub>	65.31 <sub>4.81</sub>
	Numerical RAC	41.67 <sub>5.03</sub>	42.25 <sub>5.86</sub>	46.88 <sub>5.09</sub>
	Composite Questions	82.96 <sub>3.24</sub>	71.00 <sub>4.54</sub>	68.92 <sub>3.80</sub>
	Average	68.96 <sub>1.71</sub>	59.51 <sub>1.92</sub>	67.35 <sub>1.74</sub>
19	Fluent Tracking	73.26 <sub>4.77</sub>	63.38 <sub>5.72</sub>	71.08 <sub>4.98</sub>
	State Tracking	91.01 <sub>3.03</sub>	63.64 <sub>5.13</sub>	84.27 <sub>3.86</sub>
	Action Executability	62.86 <sub>4.72</sub>	54.55 <sub>5.00</sub>	62.86 <sub>4.72</sub>
	Effects of Actions	69.57 <sub>4.80</sub>	56.63 <sub>5.44</sub>	66.67 <sub>4.89</sub>
	Numerical RAC	56.99 <sub>5.13</sub>	49.15 <sub>6.51</sub>	48.39 <sub>5.18</sub>
	Composite Questions	65.57 <sub>2.88</sub>	57.31 <sub>3.11</sub>	62.04 <sub>2.93</sub>
	Average	68.56 <sub>1.71</sub>	57.58 <sub>1.93</sub>	64.72 <sub>1.76</sub>

Table 6: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, and Llama-3.1-70B-Instruct, on the binary subset (True/False) of the benchmark, evaluated without the ramifications constraints using the Few-shot-3 setting. The results are categorized up by the action-sequence lengths and question categories.

5, while the paraphrased version scored 4.5, indicating the high effectiveness of Llama-3.1-70B-Instruct in enhancing the fluency of the data. We would like to point out that the annotators participated on a voluntary basis and were informed beforehand that no financial compensation would be provided for their contribution.

## E FINE-TUNING DETAILS

In this section, we describe the fine-tuning performed on the training split of ACTIONREASONING-BENCH described in section 3.5. We fine-tuned Llama-3.1-8B separately for binary (true/false) and free answer questions, using 6 epochs for the former and 18 epochs for the latter. The AdamW optimizer was used, with a batch size of 4 and gradient accumulation steps set to 8 for both of the training setups. Due to the available compute resources, we were limited to a maximum context length of 4096 tokens. This leaves us roughly with 27k samples for binary answers and 14.4k samples for free answer. Tables 12, 13 and 14 show the statistics of the training set that we used to train the models.

Action Seq.	Ques Categories	GPT-4o	Llama-8B-Inst	Llama-70B-Inst
1	Fluent Tracking	76.92 <sub>5.84</sub>	50.00 <sub>6.93</sub>	75.00 <sub>6.00</sub>
	State Tracking	73.33 <sub>6.59</sub>	44.44 <sub>7.41</sub>	73.68 <sub>7.14</sub>
	Action Executability	56.25 <sub>7.16</sub>	14.58 <sub>5.09</sub>	38.30 <sub>7.09</sub>
	Effects of Actions	80.00 <sub>6.32</sub>	35.00 <sub>7.54</sub>	63.64 <sub>8.37</sub>
	Numerical RAC	08.89 <sub>4.24</sub>	04.44 <sub>3.07</sub>	13.33 <sub>5.07</sub>
	Composite Questions	49.26 <sub>3.51</sub>	41.38 <sub>3.46</sub>	55.61 <sub>3.63</sub>
	Average	54.50 <sub>2.39</sub>	35.33 <sub>2.30</sub>	53.73 <sub>2.49</sub>
10	Fluent Tracking	74.00 <sub>6.20</sub>	24.00 <sub>6.04</sub>	64.00 <sub>6.79</sub>
	State Tracking	81.40 <sub>5.93</sub>	34.88 <sub>7.27</sub>	65.71 <sub>8.02</sub>
	Action Executability	50.00 <sub>7.54</sub>	09.09 <sub>4.33</sub>	34.09 <sub>7.15</sub>
	Effects of Actions	65.22 <sub>7.02</sub>	41.30 <sub>7.26</sub>	62.86 <sub>8.17</sub>
	Numerical RAC	14.29 <sub>5.00</sub>	10.20 <sub>4.32</sub>	12.24 <sub>4.68</sub>
	Composite Questions	43.84 <sub>3.48</sub>	37.93 <sub>3.41</sub>	48.13 <sub>3.65</sub>
	Average	50.57 <sub>2.40</sub>	30.34 <sub>2.20</sub>	47.00 <sub>2.50</sub>
19	Fluent Tracking	74.42 <sub>6.65</sub>	44.19 <sub>7.57</sub>	65.12 <sub>7.27</sub>
	State Tracking	61.22 <sub>6.96</sub>	28.57 <sub>6.45</sub>	54.55 <sub>7.51</sub>
	Action Executability	47.92 <sub>7.21</sub>	10.42 <sub>4.41</sub>	25.00 <sub>6.25</sub>
	Effects of Actions	73.91 <sub>6.47</sub>	27.66 <sub>6.52</sub>	60.98 <sub>7.62</sub>
	Numerical RAC	06.12 <sub>3.42</sub>	06.12 <sub>3.42</sub>	00.00 <sub>0.00</sub>
	Composite Questions	38.19 <sub>3.44</sub>	26.13 <sub>3.11</sub>	46.2 <sub>3.68</sub>
	Average	45.62 <sub>2.39</sub>	24.37 <sub>2.06</sub>	42.54 <sub>2.44</sub>

Table 7: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, and Llama-3.1-70B-Instruct, on the free answer subset of the benchmark, evaluated without the ramifications constraints using the Few-shot-3 setting. The results are categorized up by the action-sequence lengths and question categories.

Act. Seq.	Ques Categories	Blocksworld	Depots	Driverlog	Grippers	Mystery	Satellite	Spanner	Visitall
1	Fluent Tracking	100.0 <sub>0.0</sub>	92.31 <sub>7.69</sub>	69.23 <sub>13.32</sub>	95.24 <sub>4.76</sub>	100.0 <sub>0.0</sub>	88.24 <sub>8.05</sub>	88.89 <sub>7.62</sub>	100.0 <sub>0.0</sub>
	State Tracking	83.33 <sub>9.04</sub>	93.33 <sub>6.67</sub>	66.67 <sub>16.67</sub>	93.75 <sub>6.25</sub>	85.71 <sub>9.71</sub>	61.54 <sub>14.04</sub>	62.51 <sub>2.5</sub>	100.0 <sub>0.0</sub>
	Action Executability	71.43 <sub>10.1</sub>	76.47 <sub>10.6</sub>	76.47 <sub>10.6</sub>	78.95 <sub>9.61</sub>	66.67 <sub>11.43</sub>	76.19 <sub>9.52</sub>	80.09 <sub>1.8</sub>	47.06 <sub>12.48</sub>
	Effects of Act.	82.61 <sub>8.08</sub>	66.67 <sub>12.6</sub>	85.71 <sub>9.71</sub>	77.27 <sub>9.14</sub>	100.0 <sub>0.0</sub>	60.01 <sub>3.09</sub>	80.01 <sub>0.69</sub>	87.58 <sub>5.4</sub>
	Numerical RAC	64.71 <sub>11.95</sub>	53.85 <sub>14.39</sub>	46.67 <sub>13.33</sub>	50.01 <sub>3.87</sub>	35.71 <sub>13.29</sub>	44.44 <sub>17.57</sub>	31.58 <sub>10.96</sub>	31.82 <sub>10.16</sub>
	Composite	78.57 <sub>6.41</sub>	67.57 <sub>5.5</sub>	68.42 <sub>7.64</sub>	70.73 <sub>7.19</sub>	66.67 <sub>7.65</sub>	53.89 <sub>8.09</sub>	60.98 <sub>7.71</sub>	78.38 <sub>6.86</sub>
	Average	78.95 <sub>3.55</sub>	73.45 <sub>4.17</sub>	68.87 <sub>4.52</sub>	77.44 <sub>3.64</sub>	73.21 <sub>4.2</sub>	64.04 <sub>4.51</sub>	65.89 <sub>4.19</sub>	70.69 <sub>4.24</sub>
10	Fluent Tracking	75.0 <sub>25.0</sub>	91.67 <sub>8.33</sub>	100.0 <sub>0.0</sub>	100.0 <sub>0.0</sub>	80.01 <sub>3.33</sub>	73.68 <sub>10.38</sub>	76.19 <sub>9.52</sub>	94.74 <sub>5.26</sub>
	State Tracking	86.67 <sub>9.09</sub>	63.64 <sub>15.21</sub>	83.33 <sub>11.24</sub>	73.33 <sub>11.82</sub>	63.64 <sub>15.21</sub>	61.54 <sub>14.04</sub>	81.25 <sub>10.08</sub>	87.51 <sub>2.5</sub>
	Action Executability	52.38 <sub>11.17</sub>	64.29 <sub>13.29</sub>	73.68 <sub>10.38</sub>	57.14 <sub>11.07</sub>	46.67 <sub>13.33</sub>	64.71 <sub>11.95</sub>	65.01 <sub>0.94</sub>	55.01 <sub>1.41</sub>
	Effects of Act.	68.09 <sub>5.2</sub>	66.67 <sub>10.54</sub>	78.57 <sub>11.38</sub>	69.57 <sub>9.81</sub>	78.57 <sub>11.38</sub>	30.77 <sub>13.32</sub>	61.54 <sub>14.04</sub>	66.67 <sub>10.54</sub>
	Numerical RAC	57.14 <sub>11.07</sub>	44.01 <sub>10.13</sub>	43.75 <sub>12.81</sub>	37.51 <sub>18.3</sub>	33.33 <sub>12.6</sub>	42.86 <sub>11.07</sub>	29.41 <sub>11.39</sub>	27.27 <sub>9.72</sub>
	Composite	71.74 <sub>6.71</sub>	78.05 <sub>6.54</sub>	74.42 <sub>6.73</sub>	77.55 <sub>6.02</sub>	55.56 <sub>8.4</sub>	54.35 <sub>7.43</sub>	64.29 <sub>7.48</sub>	71.43 <sub>7.75</sub>
	Average	67.42 <sub>4.09</sub>	67.74 <sub>4.21</sub>	74.36 <sub>4.05</sub>	73.91 <sub>3.75</sub>	57.43 <sub>4.94</sub>	55.04 <sub>4.4</sub>	63.57 <sub>4.25</sub>	64.84 <sub>2.29</sub>
19	Fluent Tracking	93.75 <sub>6.25</sub>	66.67 <sub>12.6</sub>	62.51 <sub>8.3</sub>	100.0 <sub>0.0</sub>	71.43 <sub>18.44</sub>	93.33 <sub>6.67</sub>	85.71 <sub>9.71</sub>	61.54 <sub>14.04</sub>
	State Tracking	91.67 <sub>5.76</sub>	81.25 <sub>10.08</sub>	86.67 <sub>9.09</sub>	70.01 <sub>5.28</sub>	87.58 <sub>5.4</sub>	58.33 <sub>14.86</sub>	78.57 <sub>11.38</sub>	83.33 <sub>16.67</sub>
	Action Executability	52.17 <sub>10.65</sub>	64.29 <sub>13.29</sub>	57.89 <sub>11.64</sub>	70.01 <sub>5.1</sub>	45.45 <sub>10.87</sub>	56.25 <sub>12.81</sub>	58.82 <sub>12.3</sub>	68.18 <sub>10.16</sub>
	Effects of Act.	72.73 <sub>9.72</sub>	80.95 <sub>8.78</sub>	85.71 <sub>9.71</sub>	70.01 <sub>5.1</sub>	76.92 <sub>12.16</sub>	41.67 <sub>14.86</sub>	66.67 <sub>11.43</sub>	76.47 <sub>10.6</sub>
	Numerical RAC	50.01 <sub>1.47</sub>	57.89 <sub>11.64</sub>	38.46 <sub>14.04</sub>	30.01 <sub>0.51</sub>	12.58 <sub>5.4</sub>	37.51 <sub>2.5</sub>	26.67 <sub>11.82</sub>	31.82 <sub>10.16</sub>
	Composite	72.92 <sub>6.48</sub>	60.47 <sub>7.54</sub>	72.09 <sub>6.92</sub>	79.55 <sub>6.15</sub>	53.85 <sub>8.09</sub>	55.17 <sub>1.18</sub>	67.44 <sub>7.23</sub>	86.21 <sub>6.52</sub>
	Average	71.93 <sub>3.65</sub>	67.19 <sub>4.17</sub>	68.75 <sub>4.4</sub>	70.31 <sub>4.05</sub>	54.87 <sub>4.7</sub>	56.67 <sub>4.54</sub>	64.46 <sub>4.37</sub>	66.97 <sub>4.53</sub>

Table 8: Performance across domains on GPT-4o on both the binary and free-answer subsets of the benchmark, evaluated without the ramifications constraints using the zero-shot setting. The results are categorized by the action-sequence lengths and question categories.

Action Seq.	Fluent Types	GPT-4o	Llama-8B-Inst	Llama-70B-Inst	Finetuned Llama-8B
1	Base Fluents	81.25 <sub>5.63</sub>	65.22 <sub>7.02</sub>	74.55 <sub>5.87</sub>	96.32 <sub>2.57</sub>
	Derived Fluents	82.81 <sub>4.72</sub>	76.36 <sub>5.73</sub>	80.28 <sub>4.72</sub>	86.79 <sub>3.72</sub>
	Self-Derived Fluents	85.87 <sub>3.63</sub>	68.24 <sub>5.05</sub>	81.44 <sub>3.95</sub>	98.04 <sub>1.37</sub>
	Static Properties	85.19 <sub>6.84</sub>	72.73 <sub>7.75</sub>	80.06 <sub>6.32</sub>	89.47 <sub>4.06</sub>
10	Base Fluents	88.89 <sub>4.68</sub>	78.79 <sub>7.12</sub>	75.93 <sub>5.82</sub>	90.57 <sub>4.02</sub>
	Derived Fluents	75.31 <sub>4.79</sub>	55.38 <sub>6.17</sub>	72.73 <sub>5.08</sub>	79.38 <sub>4.11</sub>
	Self-Derived Fluents	77.66 <sub>4.3</sub>	74.29 <sub>5.22</sub>	69.39 <sub>4.66</sub>	95.15 <sub>2.12</sub>
	Static Properties	79.41 <sub>6.93</sub>	63.16 <sub>7.83</sub>	69.77 <sub>7.0</sub>	94.52 <sub>2.66</sub>
19	Base Fluents	90.24 <sub>4.63</sub>	63.64 <sub>8.37</sub>	76.19 <sub>6.57</sub>	83.61 <sub>4.74</sub>
	Derived Fluents	74.42 <sub>4.7</sub>	44.07 <sub>6.46</sub>	67.47 <sub>5.14</sub>	76.47 <sub>4.2</sub>
	Self-Derived Fluents	80.04 <sub>3.34</sub>	70.91 <sub>6.12</sub>	71.74 <sub>4.69</sub>	93.46 <sub>2.39</sub>
	Static Properties	82.76 <sub>7.01</sub>	60.71 <sub>9.23</sub>	67.57 <sub>7.41</sub>	96.88 <sub>2.17</sub>

Table 9: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, and fine-tuned Llama-3.1-8B on the binary subset (True/False) of the benchmark evaluated without the ramifications constraints using the zero-shot-CoT. The results are categorized up by the fluent types

Action Seq.	Fluent Types	GPT-4o	Llama-8B-Inst	Llama-70B-Inst	Finetuned Llama-8B
1	Positive Fluents	83.16 <sub>3.84</sub>	80.05 <sub>5.39</sub>	79.31 <sub>4.34</sub>	72.84 <sub>4.94</sub>
	Negative Fluents	77.78 <sub>4.62</sub>	60.34 <sub>6.42</sub>	70.77 <sub>5.64</sub>	77.53 <sub>4.42</sub>
	Pos. and Neg. Fluents	81.56 <sub>2.08</sub>	65.15 <sub>2.72</sub>	78.65 <sub>2.17</sub>	93.77 <sub>1.29</sub>
10	Positive Fluents	82.47 <sub>3.86</sub>	80.77 <sub>5.47</sub>	81.16 <sub>4.71</sub>	80.04 <sub>4.34</sub>
	Negative Fluents	66.27 <sub>5.19</sub>	53.66 <sub>7.79</sub>	69.23 <sub>5.72</sub>	72.29 <sub>4.91</sub>
	Pos. and Neg. Fluents	71.32 <sub>2.32</sub>	67.42 <sub>2.88</sub>	68.21 <sub>2.43</sub>	87.74 <sub>1.73</sub>
19	Positive Fluents	79.31 <sub>4.34</sub>	67.44 <sub>7.15</sub>	72.86 <sub>5.32</sub>	84.93 <sub>4.19</sub>
	Negative Fluents	71.08 <sub>4.98</sub>	58.14 <sub>7.52</sub>	67.53 <sub>5.34</sub>	75.64 <sub>4.86</sub>
	Pos. and Neg. Fluents	70.98 <sub>2.33</sub>	54.03 <sub>3.52</sub>	67.52 <sub>2.47</sub>	86.27 <sub>1.7</sub>

Table 10: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, and fine-tuned Llama-3.1-8B on the binary subset (True/False) of the benchmark evaluated without the ramifications constraints using the zero-shot-CoT. The results are categorized up by the fluent types

Domain	Annotator 1		Annotator 2		Annotator 3		Average	
	Templated	Rephrased	Templated	Rephrased	Templated	Rephrased	Templated	Rephrased
Blocksworld	3.8	4.8	5.0	4.6	3.8	4.2	4.2	4.5
Depots	4.6	4.8	3.6	4.8	4.0	4.0	4.1	4.5
Driverlog	4.8	4.8	3.6	4.2	4.4	4.6	4.3	4.5
Grippers	4.6	5.0	4.0	4.8	4.6	4.8	4.4	4.9
Mystery	4.0	4.8	4.0	4.4	3.6	4.0	3.9	4.4
Satellite	4.4	4.8	4.6	4.4	4.4	4.2	4.5	4.5
Spanner	5.0	5.0	5.0	4.6	3.8	3.8	4.6	4.5
Visitall	3.6	4.2	3.8	4.6	4.0	4.2	3.8	4.3
Average	4.3	4.8	4.2	4.6	4.1	4.2	4.2	4.5

Table 11: Naturalness scores assigned by three annotators on a scale of 1 to 5, where 1 indicates completely incoherent text and 5 indicates natural-sounding questions. The table presents scores for both the templated questions and paraphrased questions.

Answer category	No of Samples
False	13,793
True	13,319
Free-Response	14,476
Total	41,588

Table 12: Data distribution used for fine-tuning, categorized by response type. The binary question responses are split into “True” and “False”. “Free-Response” indicates the count of open-ended questions in the training set.

1026  
 1027  
 1028  
 1029  
 1030  
 1031  
 1032  
 1033  
 1034  
 1035  
 1036  
 1037  
 1038  
 1039  
 1040  
 1041  
 1042  
 1043  
 1044  
 1045  
 1046  
 1047  
 1048  
 1049  
 1050  
 1051  
 1052  
 1053  
 1054  
 1055  
 1056  
 1057  
 1058  
 1059  
 1060  
 1061  
 1062  
 1063  
 1064  
 1065  
 1066  
 1067  
 1068  
 1069  
 1070  
 1071  
 1072  
 1073  
 1074  
 1075  
 1076  
 1077  
 1078  
 1079

Question category	No of Samples (Binary)	No of Samples(Free-Response)
Fluent Tracking	11,674	4,946
State Tracking	2,264	1,133
Action Executability	1,534	1,094
Effects of Actions	1,196	1,040
Numerical RAC	5,757	3,293
Composite Questions	4,687	2,970
Total	27,112	14,476

Table 13: Data distribution used for fine-tuning, categorized by Question Categories.

Domain	No of Samples (Binary)	No of Samples (Free-Response)
Blocksworld	3,540	1,978
Depots	2,799	1,394
Driverlog	3,404	1,785
Grippers	3,494	1,910
Mystery	2,825	1,636
Satellite	4,037	2,073
Spanner	3,866	1,992
Visitall	3,147	1,708
Total	27,112	14,476

Table 14: Data distribution used for fine-tuning, categorized by Domains.

## F FREE ANSWERS EVALUATION DETAILS

We evaluate the free answers using Llama-3.1-70B-Instruct. The following is the few-shot-7 prompt that we used for evaluating the responses:

### Prompt for Free Answer Evaluation with Llama-3.1-70b-Instruct

Evaluate whether the LLM response and the ground truth response are semantically the same. Examine the responses, provide reasoning for your evaluation, and then Write "True" if the responses are the same or "False" if they are different. LLM Response or Ground Truth could be "None".

Example 1:

[LLM Response]

Location f1\_4f is connected to f1\_3f, f0\_4f, and f2\_4f.

[Ground Truth]

location f1\_4f and location f0\_4f are connected, location f1\_4f and location f2\_4f are connected, location f0\_4f and location f1\_4f are connected, location f1\_3f and location f1\_4f are connected, there is a connection between location f2\_4f and location f1\_4f, there is a connection between location f1\_4f and location f1\_3f

[Reasoning for the evaluation]

all of the connections are the same

[Answer]

True

:

Example 7:

[LLM Response]

ball1 is at room2, ball2 is at room2, ball3 is at room4, ball4 is at room1, ball5 is at room1, ball6 is at room4, ball7 is being carried by robot1's lgrripper1, ball8 is at room4, robot1 is at room2, robot1's lgrripper1 is not free, robot1's rgrripper1 is not free, robot1 is carrying ball7 with lgrripper1, robot1 is carrying ball6 with rgrripper1

[Ground Truth]

ball1 is present at room2, ball2 is at room2, ball3 is present at room4, ball4 is at room1, ball5 is located at room1, ball7 is being carried by robot1's lgrripper1, ball8 is present at room4, rgrripper1 of robot1 is carrying ball6 and robot1 is present in room2

[Reasoning for the evaluation]

According to the ground truth, robot1 is carrying ball6, but LLM gives a contradictory response: that ball6 is at room4

[Answer] False

Given the examples and instructions above, evaluate the following responses:

[LLM Response]

{llm\_response}

[Ground Truth]

{ground\_truth}

Table 15 shows the results of the free-answers using automatic metrics based on semantic similarity, F1-BERTScore (Zhang et al., 2019).

Action Seq.	Ques Categories	GPT-4o	Llama-8B-Inst	Llama-70B-Inst	Finetuned Llama-8B
1	Fluent Tracking	71.76 <sub>2.11</sub>	48.52 <sub>2.84</sub>	55.21 <sub>2.85</sub>	90.22 <sub>0.95</sub>
	State Tracking	68.24 <sub>1.98</sub>	54.85 <sub>3.18</sub>	61.07 <sub>2.59</sub>	82.06 <sub>1.82</sub>
	Action Executability	56.93 <sub>3.36</sub>	49.49 <sub>2.80</sub>	57.25 <sub>3.77</sub>	75.24 <sub>2.70</sub>
	Effects of Actions	66.43 <sub>1.94</sub>	49.82 <sub>3.53</sub>	59.48 <sub>2.67</sub>	82.12 <sub>2.07</sub>
	Numerical RAC	87.07 <sub>1.61</sub>	39.40 <sub>5.05</sub>	61.94 <sub>5.30</sub>	89.52 <sub>1.36</sub>
	Composite Questions	55.03 <sub>1.65</sub>	40.59 <sub>1.52</sub>	40.62 <sub>1.77</sub>	82.58 <sub>1.17</sub>
	Average	67.55 <sub>2.10</sub>	47.11 <sub>3.15</sub>	55.92 <sub>3.15</sub>	83.62 <sub>1.67</sub>
10	Fluent Tracking	66.46 <sub>2.29</sub>	51.24 <sub>2.98</sub>	56.89 <sub>3.20</sub>	85.40 <sub>1.74</sub>
	State Tracking	70.09 <sub>1.26</sub>	60.43 <sub>2.52</sub>	64.10 <sub>2.23</sub>	83.78 <sub>1.49</sub>
	Action Executability	61.31 <sub>3.51</sub>	54.14 <sub>2.79</sub>	68.28 <sub>2.78</sub>	79.04 <sub>2.40</sub>
	Effects of Actions	66.19 <sub>1.34</sub>	52.99 <sub>3.09</sub>	62.05 <sub>2.16</sub>	82.03 <sub>1.78</sub>
	Numerical RAC	82.38 <sub>1.82</sub>	29.14 <sub>4.13</sub>	50.39 <sub>5.46</sub>	85.20 <sub>2.62</sub>
	Composite Questions	54.57 <sub>1.68</sub>	45.67 <sub>1.59</sub>	54.41 <sub>1.39</sub>	79.83 <sub>1.56</sub>
	Average	66.83 <sub>1.98</sub>	48.93 <sub>2.85</sub>	59.35 <sub>2.87</sub>	82.47 <sub>1.93</sub>
19	Fluent Tracking	60.78 <sub>3.19</sub>	47.69 <sub>2.71</sub>	56.35 <sub>3.12</sub>	87.79 <sub>1.39</sub>
	State Tracking	70.33 <sub>1.00</sub>	58.06 <sub>2.77</sub>	62.06 <sub>2.47</sub>	83.51 <sub>1.41</sub>
	Action Executability	61.72 <sub>3.41</sub>	49.40 <sub>2.93</sub>	56.90 <sub>3.48</sub>	78.84 <sub>2.40</sub>
	Effects of Actions	65.71 <sub>1.99</sub>	52.53 <sub>3.51</sub>	62.78 <sub>3.22</sub>	84.90 <sub>1.20</sub>
	Numerical RAC	81.30 <sub>2.29</sub>	32.55 <sub>4.31</sub>	53.79 <sub>5.36</sub>	80.75 <sub>3.62</sub>
	Composite Questions	55.48 <sub>1.55</sub>	50.87 <sub>1.40</sub>	52.75 <sub>1.63</sub>	77.10 <sub>1.76</sub>
	Average	65.88 <sub>2.23</sub>	48.51 <sub>1.95</sub>	57.43 <sub>3.21</sub>	82.14 <sub>1.96</sub>

Table 15: F1-BERTScore of GPT-4o, Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, and finetuned Llama-3.1-8B on the free-answer subset of the benchmark, evaluated without the ramifications constraints using the zero-shot-CoT. The results are categorized up by the action-sequence lengths and question categories.

Below we share some examples where F1-BertScore fails to give an accurate score.

#### Low F1-BERTScore but the response is correct

[GROUND TRUTH]  
robot1’s rgripper1 is available

[RESPONSE]  
rgripper1 is free.

[BERT SCORE]  
0.71

[Llama-3.1-70B-Instruct Evaluation]

The LLM response and the ground truth response are semantically the same. The term “free” in the LLM response is equivalent to “available” in the ground truth response, both indicating that the rgripper1 is not being used. True.

[ANALYSIS]

The response is a simple paraphrase of the ground truth which is easily explained by the LLM evaluation.

1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241

### High F1-BERTScore but the response is incorrect

#### [GROUND TRUTH]

loc\_x0\_y1 is not connected to loc\_x2\_y4  
loc\_x0\_y3 is not connected to loc\_x2\_y4  
loc\_x1\_y0 and loc\_x2\_y4 are not connected  
loc\_x1\_y1 is not connected to loc\_x2\_y4  
:  
there is no connection between loc\_x3\_y0 and loc\_x2\_y4  
there is no connection between loc\_x3\_y1 and loc\_x2\_y4  
there is no connection between loc\_x3\_y2 and loc\_x2\_y4  
there is no connection between loc\_x3\_y3 and loc\_x2\_y4

#### [RESPONSE]

- loc\_x2\_y4 is not visited.  
- loc\_x2\_y4 is connected to loc\_x1\_y4.  
- loc\_x2\_y4 is connected to loc\_x3\_y4.  
- loc\_x2\_y4 is not connected to loc\_x2\_y3.  
- loc\_x2\_y4 is not connected to loc\_x2\_y5.  
:  
- loc\_x2\_y4 is not connected to loc\_x1\_y2.  
- loc\_x2\_y4 is not connected to loc\_x1\_y3.  
- loc\_x2\_y4 is not connected to loc\_x1\_y5.

#### [BERT SCORE]

0.89

#### [Llama-3.1-70B-Instruct Evaluation]

The LLM response and the ground truth response are not semantically the same. The LLM response states that loc\_x2\_y4 is not connected to loc\_x2\_y3, but the ground truth response states that loc\_x2\_y4 and loc\_x2\_y3 are connected. Additionally, the LLM response states that loc\_x2\_y4 is not connected to loc\_x2\_y5, but the ground truth response states that loc\_x2\_y4 and loc\_x2\_y5 are connected. False.

#### [ANALYSIS]

The response contains a lot of incorrect responses. Moreover, the response fails to mention the location of the robot. The Llama-3.1-70B-Instruct evaluation also fails to mention the lack of the robot but correctly identifies some incorrect connections.

## G DOMAINS IN ACTIONREASONINGBENCH

In the following sub-sections, we provide details regarding all the domains used in our study. We first present the PDDL description that is given in the IPC and then present how state-space calculation is performed for that domain. State space represents the possible number of interactions that can be performed at a particular state. A higher state-space represents a more difficult problem for traditional AI solvers.

### G.1 BLOCKSWORLD

In this domain, we have a set of blocks that can be manipulated using four basic actions: picking up a block from the table, putting down a block onto the table, stacking one block onto another, and unstacking a block from atop another block. The goal is to move and stack these blocks using a robotic hand, following specific rules and conditions.

This domain is formally represented in the IPC using the PDDL as outlined below:

Domain Name	# Fluents Defined	# Actions Defined	# Object Defined	IPC Year	State Complexity
<b>Blocksworld</b>	5	4	1	2000	$O(2^{N^2+2N})$
<b>Depots</b>	6	5	6	2002	$O(2^{4N^2+2N})$
<b>DriverLog</b>	5	6	4	2002	$O(2^{5N^2-N})$
<b>Grippers</b>	4	3	4	1998	$O(2^{N^3+3N^2})$
<b>Mystery</b>	7	3	5	1998	$O(2^{7N^2-3N})$
<b>Satellite</b>	8	5	4	2002	$O(2^{5N^2+3N})$
<b>Spanner</b>	6	3	4	2011	$O(2^{3N^2+2N})$
<b>Visitall</b>	3	1	1	2014	$O(2^{N^2+N})$

Table 16: Summarizing key characteristics of various domains in ACTIONREASONINGBENCH including the number of fluents, actions, and objects defined within each domain. Domains are categorized by year of introduction in the IPC and state space complexity, which reflects the difficulty level for AI planners to solve each domain. A larger state space typically indicates greater complexity and presents more significant challenges for traditional AI planners. “ $N$ ” represents the number of objects in each instance. For example, in the *Spanner* domain,  $N$  refers to the number of locations, spanners and nuts.

#### PDDL description for `Blocksworld` domain

```
(define (domain BLOCKS)
  (:requirements :strips :typing)
  (:types block)
  (:predicates (on ?x - block ?y - block)
    (ontable ?x - block)
    (clear ?x - block)
    (handempty)
    (holding ?x - block))

  (:action pick-up
    :parameters (?x - block)
    :precondition (and (clear ?x) (ontable ?x) (handempty))
    :effect
    (and (not (ontable ?x))
      (not (clear ?x))
      (not (handempty))
      (holding ?x)))
```



```

1296
1297 (:action put-down
1298 :parameters (?x - block)
1299 :precondition (holding ?x)
1300 :effect
1301 (and (not (holding ?x))
1302 (clear ?x)
1303 (handempty)
1304 (ontable ?x)))
1305
1306 (:action stack
1307 :parameters (?x - block ?y - block)
1308 :precondition (and (holding ?x) (clear ?y))
1309 :effect
1310 (and (not (holding ?x))
1311 (not (clear ?y))
1312 (clear ?x)
1313 (handempty)
1314 (on ?x ?y)))
1315
1316 (:action unstack
1317 :parameters (?x - block ?y - block)
1318 :precondition (and (on ?x ?y) (clear ?x) (handempty))
1319 :effect
1320 (and (holding ?x)
1321 (clear ?y)
1322 (not (clear ?x))
1323 (not (handempty))
1324 (not (on ?x ?y))))
1325 )

```

Predicates define the number of fluents in a state:

- on(b1,b2)
- ontable(b)
- clear(b)
- holding(b)
- handempty

The complexity of a state is, where  $N$  is the number of objects

$$O(2^{N^2+2N}) \tag{1}$$

## G.2 DEPOTS

The Depots domain models a logistics environment where crates are transported between different locations using trucks and manipulated using hoists. The goal is to efficiently move crates from one location to another, utilizing the available resources (hoists and trucks) while adhering to the constraints defined by the predicates and actions.

This domain is formally represented in the IPC using the PDDL as outlined below:

```

1344 PDDL description for Depots domain
1345
1346 (define (domain depots)
1347 (:requirements :strips :typing)
1348 (:types place locatable - object
1349 depot distributor - place

```

```

1350 truck hoist surface - locatable
1351 pallet crate - surface)
1352
1353 (:predicates (at ?x - locatable ?y - place)
1354 (on ?x - crate ?y - surface)
1355 (in ?x - crate ?y - truck)
1356 (lifting ?x - hoist ?y - crate)
1357 (available ?x - hoist)
1358 (clear ?x - surface))
1359
1360 (:action Drive
1361 :parameters (?x - truck ?y - place ?z - place)
1362 :precondition (and (at ?x ?y))
1363 :effect (and (not (at ?x ?y)) (at ?x ?z)))
1364
1365 (:action Lift
1366 :parameters (?x - hoist ?y - crate ?z - surface ?p - place)
1367 :precondition (and (at ?x ?p) (available ?x) (at ?y ?p) (on ?y ?z) (clear ?y))
1368 :effect (and (not (at ?y ?p)) (lifting ?x ?y) (not (clear ?y)) (not (available ?x)) (clear ?z) (not
1369 (on ?y ?z))))
1370
1371 (:action Drop
1372 :parameters (?x - hoist ?y - crate ?z - surface ?p - place)
1373 :precondition (and (at ?x ?p) (at ?z ?p) (clear ?z) (lifting ?x ?y))
1374 :effect (and (available ?x) (not (lifting ?x ?y)) (at ?y ?p) (not (clear ?z)) (clear ?y)(on ?y
1375 ?z)))
1376
1377 (:action Load
1378 :parameters (?x - hoist ?y - crate ?z - truck ?p - place)
1379 :precondition (and (at ?x ?p) (at ?z ?p) (lifting ?x ?y))
1380 :effect (and (not (lifting ?x ?y)) (in ?y ?z) (available ?x)))
1381
1382 (:action Unload
1383 :parameters (?x - hoist ?y - crate ?z - truck ?p - place)
1384 :precondition (and (at ?x ?p) (at ?z ?p) (available ?x) (in ?y ?z))
1385 :effect (and (not (in ?y ?z)) (not (available ?x)) (lifting ?x ?y)))
1386 )

```

Predicates define the number of fluents in a state:

- (at ?x - locatable ?y - place)
- (on ?x - crate ?y - surface)
- (in ?x - crate ?y - truck)
- (lifting ?x - hoist ?y - crate)
- (available ?x - hoist)
- (clear ?x - surface))

$$O(2^{4N^2+2N}) \quad (2)$$

### G.3 DRIVERLOG

This domain is modeled to simulate logistics operations where drivers, trucks, and objects must be moved between different locations. The primary focus is on transporting objects via trucks, either driven by drivers or moved manually by walking. The key actions in this domain include loading and unloading trucks, drivers boarding and disembarking trucks, driving trucks between connected locations, and walking when no truck is involved.

1404 This domain is formally represented in the IPC using the PDDL as outlined below:  
 1405  
 1406

1407 **PDDL description for `Driverlog` domain**  
 1408

```

1409 (define (domain driverlog)
1410 (:requirements :typing) (:types location locatable - object
1411 driver truck obj - locatable)
1412 (:predicates
1413 (at ?obj - locatable ?loc - location)
1414 (in ?obj1 - obj ?obj - truck)
1415 (driving ?d - driver ?v - truck)
1416 (link ?x ?y - location) (path ?x ?y - location)
1417 (empty ?v - truck))

1418 (:action LOAD-TRUCK
1419 :parameters
1420 (?obj - obj
1421 ?truck - truck
1422 ?loc - location)
1423 :precondition (and (at ?truck ?loc) (at ?obj ?loc))
1424 :effect (and (not (at ?obj ?loc)) (in ?obj ?truck)))

1425 (:action UNLOAD-TRUCK
1426 :parameters
1427 (?obj - obj
1428 ?truck - truck
1429 ?loc - location)
1430 :precondition (and (at ?truck ?loc) (in ?obj ?truck))
1431 :effect (and (not (in ?obj ?truck)) (at ?obj ?loc)))

1432 (:action BOARD-TRUCK
1433 :parameters
1434 (?driver - driver
1435 ?truck - truck
1436 ?loc - location)
1437 :precondition (and (at ?truck ?loc) (at ?driver ?loc) (empty ?truck))
1438 :effect (and (not (at ?driver ?loc)) (driving ?driver ?truck) (not (empty ?truck))))

1439 (:action DISEMBARK-TRUCK
1440 :parameters
1441 (?driver - driver
1442 ?truck - truck
1443 ?loc - location)
1444 :precondition (and (at ?truck ?loc) (driving ?driver ?truck))
1445 :effect (and (not (driving ?driver ?truck)) (at ?driver ?loc) (empty ?truck)))

1446 (:action DRIVE-TRUCK
1447 :parameters
1448 (?truck - truck
1449 ?loc-from - location
1450 ?loc-to - location
1451 ?driver - driver)
1452 :precondition
1453 (and (at ?truck ?loc-from)
1454 (driving ?driver ?truck) (link ?loc-from ?loc-to))
1455 :effect (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))
1456
1457
```

```

1458 (:action WALK
1459 :parameters
1460 (?driver - driver
1461 ?loc-from - location
1462 ?loc-to - location)
1463 :precondition (and (at ?driver ?loc-from) (path ?loc-from ?loc-to))
1464 :effect (and (not (at ?driver ?loc-from)) (at ?driver ?loc-to)))
1465 )
1466

```

Predicates define the number of fluents in a state:

- (at ?obj - locatable ?loc - location)
- (in ?obj1 - obj ?obj - truck)
- (driving ?d - driver ?v - truck)
- (link ?x ?y - location)
- (path ?x ?y - location)
- (empty ?v - truck)

$$O(2^{5N^2-N}) \quad (3)$$

#### G.4 GRIPPER

This domain represents a transportation domain where a robot with two grippers can move between rooms, pick up objects, and drop them off. The robot can only hold one object in each gripper at a time. This domain could solve tasks where the robot needs to transport multiple objects from one room to another by strategically moving, picking up, and dropping items.

This domain is formally represented in the IPC using the PDDL as outlined below:

```

1491 PDDL description for Gripper domain
1492
1493 (define (domain gripper-strips)
1494 (:requirements :strips :typing)
1495 (:types room object robot gripper)
1496 (:predicates (at-robby ?r - robot ?x - room)
1497 (at ?o - object ?x - room)
1498 (free ?r - robot ?g - gripper)
1499 (carry ?r - robot ?o - object ?g - gripper))
1500
1501 (:action move
1502 :parameters (?r - robot ?from ?to - room)
1503 :precondition (and (at-robby ?r ?from))
1504 :effect (and (at-robby ?r ?to) (not (at-robby ?r ?from))))
1505
1506 (:action pick
1507 :parameters (?r - robot ?obj - object ?room - room ?g - gripper)
1508 :precondition (and (at ?obj ?room) (at-robby ?r ?room) (free ?r ?g))
1509 :effect (and (carry ?r ?obj ?g)
1510 (not (at ?obj ?room))
1511 (not (free ?r ?g))))

```

1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565

```
(:action drop
:parameters (?r - robot ?obj - object ?room - room ?g - gripper)
:precondition (and (carry ?r ?obj ?g) (at-robby ?r ?room))
:effect (and (at ?obj ?room)
(free ?r ?g)
(not (carry ?r ?obj ?g))))
```

Predicates define the number of fluents in a state:

- (carry ?r - robot ?o - object ?g - gripper)
- (at-robby ?r - robot ?x - room)
- (at ?o - object ?x - room)
- (free ?r - robot ?g - gripper)

$$O(2^{N^3+3N^2}) \tag{4}$$

## G.5 MYSTERY

The Mystery domain represents a transportation system where vehicles move between locations, constrained by fuel levels, and can load or unload cargo, constrained by available space. The key aspects of this domain are managing fuel for vehicle movement and managing space for loading and unloading cargo. Locations are connected, and the system also handles fuel transitions, space transitions, and the movement of objects across a grid of locations.

This domain is formally represented in the IPC using the PDDL as outlined below:

```
PDDL description for Mystery domain

(define (domain mystery-strips)
  (:requirements :typing)
  (:types space fuel location movable - object
  vehicle cargo - movable)
  (:predicates
  (at ?v - movable ?l - location)
  (conn ?l1 ?l2 - location)
  (has-fuel ?l - location ?f - fuel)
  (fuel-neighbor ?f1 ?f2 - fuel)
  (in ?c - cargo ?v - vehicle)
  (has-space ?v - vehicle ?s - space)
  (space-neighbor ?s1 ?s2 - space))

  (:action move
  :parameters (?v - vehicle ?l1 ?l2 - location ?f1 ?f2 - fuel)
  :precondition (and (at ?v ?l1)
  (conn ?l1 ?l2)
  (has-fuel ?l1 ?f1)
  (fuel-neighbor ?f2 ?f1))
  :effect (and (not (at ?v ?l1))
  (at ?v ?l2)
  (not (has-fuel ?l1 ?f1))
  (has-fuel ?l1 ?f2)))
```

```

1566
1567 (:action load
1568 :parameters (?c - cargo ?v - vehicle ?l - location ?s1 ?s2 - space)
1569 :precondition (and (at ?c ?l)
1570 (at ?v ?l)
1571 (has-space ?v ?s1)
1572 (space-neighbor ?s2 ?s1))
1573 :effect (and (not (at ?c ?l))
1574 (in ?c ?v)
1575 (not (has-space ?v ?s1))
1576 (has-space ?v ?s2)))
1577
1577 (:action unload
1578 :parameters (?c - cargo ?v - vehicle ?l - location ?s1 ?s2 - space)
1579 :precondition (and (in ?c ?v)
1580 (at ?v ?l)
1581 (has-space ?v ?s1)
1582 (space-neighbor ?s1 ?s2))
1583 :effect (and (not (in ?c ?v))
1584 (at ?c ?l)
1585 (not (has-space ?v ?s1))
1586 (has-space ?v ?s2)))
1587 )

```

Predicates define the number of fluents in a state:

- (at ?v - movable ?l - location)
- (has-fuel ?l - location ?f - fuel)
- (in ?c - cargo ?v - vehicle)
- (has-space ?v - vehicle ?s - space)
- (conn ?l1 ?l2 - location)
- (fuel-neighbor ?f1 ?f2 - fuel)
- (space-neighbor ?s1 ?s2 - space)

$$O(2^{7N^2-3N}) \quad (5)$$

## 1607 G.6 SATELLITE

1608 The Satellite domain represents a simplified model for managing and controlling satellites and their  
 1609 onboard instruments. The goal in this domain is to coordinate the behavior of satellites, including  
 1610 turning them toward desired directions, powering instruments on and off, calibrating instruments,  
 1611 and capturing images.  
 1612

1613 This domain is formally represented in the IPC using the PDDL as outlined below:  
 1614

1615 PDDL description for `Satellite` domain

```

1616
1617 (define (domain satellite)
1618 (:requirements :strips :typing)
1619 (:types satellite direction instrument mode)

```

```

1620 (:predicates
1621 (on_board ?i - instrument ?s - satellite)
1622 (supports ?i - instrument ?m - mode)
1623 (pointing ?s - satellite ?d - direction)
1624 (power_avail ?s - satellite)
1625 (power_on ?i - instrument)
1626 (calibrated ?i - instrument)
1627 (have_image ?d - direction ?m - mode)
1628 (calibration_target ?i - instrument ?d - direction))
1629
1630 (:action turn_to
1631 :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
1632 :precondition (and (pointing ?s ?d_prev))
1633 :effect (and (pointing ?s ?d_new) (not (pointing ?s ?d_prev))))
1634
1635 (:action switch_on
1636 :parameters (?i - instrument ?s - satellite)
1637 :precondition (and (on_board ?i ?s) (power_avail ?s))
1638 :effect (and (power_on ?i) (not (calibrated ?i)) (not (power_avail ?s))))
1639
1640 (:action switch_off
1641 :parameters (?i - instrument ?s - satellite)
1642 :precondition (and (on_board ?i ?s) (power_on ?i))
1643 :effect (and (not (power_on ?i)) (power_avail ?s)))
1644
1645 (:action calibrate
1646 :parameters (?s - satellite ?i - instrument ?d - direction)
1647 :precondition (and (on_board ?i ?s)
1648 (calibration_target ?i ?d)
1649 (pointing ?s ?d)
1650 (power_on ?i))
1651 :effect (calibrated ?i))
1652
1653 (:action take_image
1654 :parameters (?s - satellite ?d - direction ?i - instrument ?m - mode)
1655 :precondition (and (calibrated ?i)
1656 (on_board ?i ?s)
1657 (supports ?i ?m)
1658 (power_on ?i)
1659 (pointing ?s ?d))
1660 :effect (have_image ?d ?m)))

```

Predicates define the number of fluents in a state:

- 1661 • (on-board ?i - instrument ?s - satellite)
- 1662
- 1663 • (supports ?i - instrument ?m - mode)
- 1664
- 1665 • (pointing ?s - satellite ?d - direction)
- 1666
- 1667 • (have-image ?d - direction ?m - mode)
- 1668
- 1669 • (calibration-target ?i - instrument ?d - direction)
- 1670
- 1671 • (power-avail ?s - satellite)
- 1672
- 1673 • (power-on ?i - instrument)
- (calibrated ?i - instrument)

$$O(2^{5N^2+3N}) \quad (6)$$

1674 G.7 SPANNER  
1675

1676 This domain models a simple world where a man moves between locations, picks up spanners, and  
1677 uses them to tighten loose nuts. The actions available to the man involve walking between locations,  
1678 picking up the spanner, and tightening nuts using the spanner if all conditions are met.

1679 This domain is formally represented in the IPC using the PDDL as outlined below:  
1680

```

1681 PDDL description for Spanner domain
1682
1683 (define (domain spanner)
1684 (:requirements :typing :strips)
1685 (:types
1686 location locatable - object
1687 man nut spanner - locatable
1688 )
1689
1690 (:predicates
1691 (at ?m - locatable ?l - location)
1692 (carrying ?m - man ?s - spanner)
1693 (useable ?s - spanner)
1694 (link ?l1 - location ?l2 - location)
1695 (tightened ?n - nut)
1696 (loose ?n - nut))
1697
1698 (:action walk
1699 :parameters (?start - location ?end - location ?m - man)
1700 :precondition (and (at ?m ?start) (link ?start ?end))
1701 :effect (and (not (at ?m ?start)) (at ?m ?end)))
1702
1703 (:action pickup_spanner
1704 :parameters (?l - location ?s - spanner ?m - man)
1705 :precondition (and (at ?m ?l) (at ?s ?l))
1706 :effect (and (not (at ?s ?l)) (carrying ?m ?s)))
1707
1708 (:action tighten_nut
1709 :parameters (?l - location ?s - spanner ?m - man ?n - nut)
1710 :precondition (and (at ?m ?l)
1711 (at ?n ?l)
1712 (carrying ?m ?s)
1713 (useable ?s)
1714 (loose ?n))
1715 :effect (and (not (loose ?n))(not (useable ?s)) (tightened ?n)))
1716 )

```

1715 Predicates define the number of fluents in a state:  
1716

- 1717 • (at ?m - locatable ?l - location)
- 1718 • (carrying ?m - man ?s - spanner)
- 1719 • (link ?l1 - location ?l2 - location)
- 1720 • (useable ?s - spanner)
- 1721 • (tightened ?n - nut)
- 1722 • (loose ?n - nut)

1723  
1724  
1725  
1726  
1727

$$O(2^{3N^2+2N}) \quad (7)$$



## 1728 G.8 VISITALL

1729  
1730 The VisitAll domain is focused on controlling a robot that needs to visit all places on a connected  
1731 grid. The robot’s movement is governed by the connectivity of places, and each move changes  
1732 the robot’s location and marks the visited place. The task is essentially to traverse the entire grid,  
1733 visiting every place, while ensuring the robot follows connectivity constraints.

1734 This domain is formally represented in the IPC using the PDDL as outlined below:

### 1735 PDDL description for `VisitAll` domain

```
1736 (define (domain grid-visit-all)
1737   (:requirements :typing)
1738   (:types place - object)
1739   (:predicates (connected ?x ?y - place)
1740     (at-robot ?x - place)
1741     (visited ?x - place))
1742
1743   (:action move
1744     :parameters (?curpos ?nextpos - place)
1745     :precondition (and (at-robot ?curpos) (connected ?curpos ?nextpos))
1746     :effect (and (at-robot ?nextpos) (not (at-robot ?curpos)) (visited ?nextpos)))
1747   )
1748 )
```

1749  
1750 Predicates define the number of fluents in a state:

- 1751 • (connected ?x ?y - place)
- 1752 • (at-robot ?x - place)
- 1753 • (visited ?x - place)

$$1754 \quad O(2^{N^2+N}) \quad (8)$$

## 1755 H PLANNING DESCRIPTION AND TOOLS

1756  
1757 Planning, at its core, involves determining a sequence of actions that transforms the world from  
1758 an initial state to a goal state. A world state specifies which fluents are true or false at any given  
1759 time. The planning domain, denoted as  $D$ , specifies the fluents, actions, and their effects within  
1760 the system. Typically, planning domains are represented using formal languages such as PDDL or  
1761 ASP. In these languages, a transition function  $\Phi_D : states \times actions \rightarrow states$  defines how actions  
1762 transform an initial state into a resulting state.

### 1763 H.1 PLANNING DOMAIN DEFINITION LANGUAGE (PDDL)

1764  
1765 PDDL is a formal language developed for expressing planning problems and domain models. Since  
1766 its inception, PDDL has been extended to address increasingly complex planning scenarios, par-  
1767 ticularly those involving deterministic problems (Haslum et al., 2019). PDDL facilitates the spec-  
1768 ification of both the planning domains and problem instances, including objects, initial, and goal  
1769 states. In this study, we employ the “STRIPS” (Stanford Research Institute Problem Solver) subset  
1770 of PDDL (Fikes & Nilsson, 1971). Additionally, the domains are “typed”, meaning that objects in  
1771 the planning problem are assigned specific types and subtypes, ensuring a structured representation  
1772 of the problem space.

### 1773 H.2 ANSWER SET PROGRAMMING (ASP)

1774  
1775 ASP is a declarative approach to problem-solving based on logic programming and non-monotonic  
1776 reasoning. Unlike traditional planning languages like PDDL, ASP focuses on defining constraints  
1777 and rules that describe potential solutions, rather than directly encoding state transitions. In ASP, a

1782 problem is encoded as a logic program consisting of rules, facts, and constraints, and the solution is  
 1783 an "answer set" that satisfies all the constraints of the problem. In this study, we use ASP to generate  
 1784 the complete state-space by applying the sequence of actions starting from the initial state.  
 1785

## 1786 I CLASSIFICATION OF FLUENTS 1787

1788 The subsequent sections provide a detailed classification of fluents across all 13 domains included  
 1789 in ACTIONREASONINGBENCH, as described in section 3.2.  
 1790

### 1791 I.1 BLOCKSWORLD 1792

1793 In the BLOCKSWORLD domain, the fluents are categorized as follows:  
 1794

- 1795 1. Static Properties - No static properties are present
- 1796 2. Base Fluents - `onTable(block)`
- 1797 3. Derived Fluents - `clear(block)`, `handEmpty`
- 1798 4. Self-Derived Fluents - `holding(block)`, `on(block, block)`  
1799

### 1800 I.2 DEPOTS 1801

1802 In the DEPOTS domain, the fluents are categorized as follows:  
 1803

- 1804 1. Static Properties - No static properties are present
- 1805 2. Base Fluents - No base fluents are present
- 1806 3. Derived Fluents - `clear(surface)`, `available(hoist)`
- 1807 4. Self-Derived Fluents - `at(locatable, place)`, `on(crate, surface)`,  
 1808 `in(crate, truck)`, `lifting(hoist, crate)`  
 1809  
 1810

### 1811 I.3 DRIVERLOG 1812

1813 In the DRIVERLOG domain, the fluents are categorized as follows:  
 1814

- 1815 1. Static Properties - `link(location, location)`, `path(location, location)`
- 1816 2. Base Fluents - No base fluents are present
- 1817 3. Derived Fluents - `empty(truck)`
- 1818 4. Self-Derived Fluents - `at(locatable, location)`, `in(object, truck)`,  
 1819 `driving(driver, truck)`  
 1820

### 1821 I.4 GRIPPERS 1822

1823 In the GRIPPERS domain, the fluents are categorized as follows:  
 1824

- 1825 1. Static Properties - No static properties are present
- 1826 2. Base Fluents - `carry(robot, object, gripper)`
- 1827 3. Derived Fluents - `free(robot, gripper)`
- 1828 4. Self-Derived Fluents - `at_robby(robot, room)`, `at(object, room)`  
 1829  
 1830

### 1831 I.5 MYSTERY 1832

1833 In the MYSTERY domain, the fluents are categorized as follows:  
 1834

- 1835 1. Static Properties - `space_neighbor(space, space)`,  
`fuel_neighbor(fuel, fuel)`, `conn(location, location)`

- 1836 2. Base Fluents - No base fluents are present  
1837  
1838 3. Derived Fluents - No derived fluents are present  
1839  
1840 4. Self-Derived Fluents - `at(movable, location)`, `in(cargo, vehicle)`,  
1841 `has_space(vehicle, space)`, `has_fuel(location, fuel)`  
1842  
1843

#### 1844 I.6 SATELLITE

1845

1846 In the SATELLITE domain, the fluents are categorized as follows:  
1847

- 1848 1. Static Properties - `on_board(instrument, satellite)`,  
1849 `supports(instrument, mode)`, `calibration_target(instrument, direction)`  
1850  
1851 2. Base Fluents - `power_on(instrument)`, `calibrated(instrument)`,  
1852 `have_image(direction, mode)`  
1853  
1854 3. Derived Fluents - `power_avail(satellite)`  
1855  
1856 4. Self-Derived Fluents - `pointing(satellite, direction)`  
1857  
1858

#### 1859 I.7 SPANNER

1860

1861 In the SPANNER domain, the fluents are categorized as follows:  
1862

- 1863 1. Static Properties - `link(location, location)`  
1864  
1865 2. Base Fluents - `carrying(man, spanner)`, `tightened(nut)`  
1866  
1867 3. Derived Fluents - `loose(nut)`, `useable(spanner)`  
1868  
1869 4. Self-Derived Fluents - `at(locatable, location)`  
1870  
1871

#### 1872 I.8 VISITALL

1873

1874 In the VISITALL domain, the fluents are categorized as follows:  
1875

- 1876 1. Static Properties - `connected(place, place)`  
1877  
1878 2. Base Fluents - No base fluents are present  
1879  
1880 3. Derived Fluents - `visited(place)`  
1881  
1882 4. Self-Derived Fluents - `at_robot(place)`  
1883  
1884  
1885

## 1886 J RESPOSNES ON RAMIFICATION DATASET

1887

1888

1889 In the following sections, we illustrate some instances and responses that we see from GPT-4o and o1-preview.

## 1890 J.1 GPT-4o

1891

1892

1893

1894

## A Ramificaiton question for State Tracking

1895

1896

1897

1898

1899

1900

1901

1902

1903

1904

## [DOMAIN DESCRIPTION]

A block can only be picked up if it is clear, on the table, and the hand is empty, resulting in the block being held. A held block can be put down, placing it back on the table. Blocks can be stacked if the first block is held and the second block is clear, causing the first block to rest on top of the second. Unstacking occurs when the hand is empty, the first block is clear, and on top of the second, resulting in the first block being held again. A block is clear if nothing is on top of it and it's not held, and the hand is empty if it's not holding anything. Blocks are stable when clear and on the table, and they can be painted if stable and the hand is empty. A block is considered on display if it can be painted and has no other block on top of it.

1905

1906

1907

1908

1909

1910

## [INITIAL CONDITIONS]

Block b1 is stacked on block b4, block b2 has no blocks on top of it, block b2 is stacked on block b6, block b3 has no blocks on top of it, block b3 is stacked on block b5, block b4 is positioned on the table, block b5 is stacked on block b7, block b6 is positioned on the table, block b7 is stacked on block b1 and the hand is empty.

1911

1912

1913

1914

1915

1916

## [QUESTION]

Given the initial condition, the following actions are taken: block b3 is unstacked from block b5 to achieve the current state. In this state, if block b3 is placed down, what are all the valid properties of the state (including both affirmative and negated properties)? If there are no valid properties, state None.

1917

1918

1919

1920

1921

## Response to the above question

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

1934

1935

1936

1937

1938

1939

1940

1941

1942

1943

## [REASON]

To determine the valid properties of the state after block b3 is placed down, we need to follow the sequence of actions and check the conditions for each property.

1. **Initial Conditions:**

- b1 is on b4.
- b2 is on b6.
- b3 is on b5.
- b4 is on the table.
- b5 is on b7.
- b6 is on the table.
- b7 is on b1.
- The hand is empty.

2. **Action: Unstack b3 from b5:**

- The hand must be empty.
- b3 must be clear and on top of b5.
- Result: b3 is held, b5 is now clear.

3. **Current State after unstacking b3:**

- b1 is on b4.
- b2 is on b6.
- b3 is held.

1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997

- b4 is on the table.
  - b5 is on b7 and is clear.
  - b6 is on the table.
  - b7 is on b1.
  - The hand is not empty (holding b3).
4. **\*\*Action: Place b3 down:\*\***
- The hand must be holding b3.
  - Result: b3 is placed on the table, the hand is empty.
5. **\*\*Final State after placing b3 down:\*\***
- b1 is on b4.
  - b2 is on b6.
  - b3 is on the table and clear.
  - b4 is on the table.
  - b5 is on b7 and is clear.
  - b6 is on the table.
  - b7 is on b1.
  - The hand is empty.
6. **\*\*Valid Properties:\*\***
- b1 is on b4.
  - b2 is on b6.
  - b3 is on the table.
  - b3 is clear.
  - b4 is on the table.
  - b5 is on b7.
  - b5 is clear.
  - b6 is on the table.
  - b7 is on b1.
  - The hand is empty.
  - b1 is not clear (b7 is on it).
  - b2 is not clear (nothing on it but it is on b6).
  - b4 is clear (nothing on it and on the table).
  - b6 is clear (nothing on it and on the table).
  - b7 is not clear (b5 is on it).
  - b3 is stable (clear and on the table).
  - b4 is stable (clear and on the table).
  - b6 is stable (clear and on the table).
  - b3 can be painted (stable and hand is empty).
  - b4 can be painted (stable and hand is empty).
  - b6 can be painted (stable and hand is empty).
  - b3 is on display (can be painted and no block on top).
  - b4 is on display (can be painted and no block on top).
  - b6 is on display (can be painted and no block on top).
- [ANSWER]
- b1 is on b4.
  - b2 is on b6.
  - b3 is on the table.
  - b3 is clear.
  - b4 is on the table.
  - b5 is on b7.
  - b5 is clear.
  - b6 is on the table.
  - b7 is on b1.

1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051

- The hand is empty.
- b1 is not clear.
- b2 is not clear.
- b4 is clear.
- b6 is clear.
- b7 is not clear.
- b3 is stable.
- b4 is stable.
- b6 is stable.
- b3 can be painted.
- b4 can be painted.
- b6 can be painted.
- b3 is on display.
- b4 is on display.
- b6 is on display.

From the above response, we can see that block b4 is **not** stable, but the model says it is stable.

## J.2 O1-PREVIEW

**A Ramificaiton question for State Tracking**

[DOMAIN DESCRIPTION]  
A block can only be picked up if it is clear, on the table, and the hand is empty, resulting in the block being held. A held block can be put down, placing it back on the table. Blocks can be stacked if the first block is held and the second block is clear, causing the first block to rest on top of the second. Unstacking occurs when the hand is empty, the first block is clear, and on top of the second, resulting in the first block being held again. A block is clear if nothing is on top of it and it's not held, and the hand is empty if it's not holding anything. Blocks are stable when clear and on the table, and they can be painted if stable and the hand is empty. A block is considered on display if it can be painted and has no other block on top of it.

[INITIAL CONDITIONS]  
Block b1 is positioned above block b7, block b2 has no blocks on it, block b2 is stacked on block b6, block b3 is stacked on block b4, block b4 is situated on the table, block b5 has no blocks on it, block b5 is situated on the table, block b6 is stacked on block b3, block b7 is on the table, block b8 is positioned above block b1, block b9 has no blocks on it, block b9 is stacked on block b8, and the hand is empty.

[QUESTION]  
Based on the initial condition, the following actions are taken: block b9 is removed from block b8 to achieve the current state. In this state, identify all valid properties that do not include negations and list them; if there are no such properties, indicate None.

2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105

Response to the above question

[REASON]  
After removing block b9 from block b8, block b9 is placed on the table, and the hand remains empty. Block b9 now has no blocks on top of it and is not held, so it is **clear**. Since it is on the table, it is also **stable**. With the hand empty and block b9 being stable, it **can be painted**. As there are no blocks on top of it, block b9 is **on display**.

Similarly, block b5 is on the table with no blocks on top and is not held, making it **clear** and **stable**. It **can be painted** and is **on display**.

Block b7 is on the table and, with no blocks explicitly on top of it after the action, is **clear**, **stable**, **can be painted**, and is **on display**.

Blocks b1 and b8 have no blocks on top of them and are not held, so they are **clear**.

Block b2 has no blocks on top and is not held, so it is **clear**, though it is not on the table and thus not stable.

The hand is explicitly stated to be **empty** in the current state.

[ANSWER]  
- The hand is empty  
- Block b1 is clear  
- Block b2 is clear  
- Block b5 is clear  
- Block b5 is on the table  
- Block b5 is stable  
- Block b5 can be painted  
- Block b5 is on display  
- Block b7 is clear  
- Block b7 is on the table  
- Block b7 is stable  
- Block b7 can be painted  
- Block b7 is on display  
- Block b8 is clear  
- Block b9 is clear  
- Block b9 is on the table  
- Block b9 is stable  
- Block b9 can be painted  
- Block b9 is on display

From the above response, we can see that block b1 is **not** clear, but the model says it is clear.

## K IMPACT OF INPUT FORMATS ON LLM PERFORMANCE

In this section, we explore the effects of different input formats on LLMs. Specifically, we compare the performance of LLMs when inputs are presented in formal language (PDDL) or in templated formats generated during the data creation pipeline (described in Section 3.4), as opposed to natural language. For our analysis, we sampled 10% of the test set from ACTIONREASONINGBENCH, covering all question categories and action sequence lengths.

### K.1 FORMAL LANGUAGE (PDDL)

Table 17 presents the performance of LLMs when provided inputs in the formal language PDDL. Our results reveal a significant drop in performance for most models. GPT-4o shows a notable performance degradation of 16.03% compared to the natural language baseline. Similarly, Llama-3.1-70B-Instruct experiences a 6.7% decrease in accuracy. Interestingly, Llama-3.1-8B-Instruct exhibits

2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159

Action Seq.	GPT-4o	Llama-8B-Inst	Llama-70B-Inst
1	43.90 <sub>7.75</sub>	26.83 <sub>6.92</sub>	53.66 <sub>7.79</sub>
10	48.78 <sub>7.81</sub>	17.07 <sub>5.88</sub>	31.71 <sub>7.27</sub>
19	33.33 <sub>7.55</sub>	17.95 <sub>6.15</sub>	25.64 <sub>6.99</sub>

Table 17: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, and Llama-3.1-70B-Instruct on the free-answer subset of the benchmark evaluated without the ramifications constraints using the zero-shot-CoT. The input is given in the formal language, i.e. PDDL. The results are categorized up by the action sequence length.

a 3.78% increase in performance, likely attributable to its initially low baseline performance. These findings suggest that the pretraining phase of LLMs, predominantly focused on natural language, plays a crucial role in shaping their reasoning capabilities. Consequently, formal language inputs, that deviate from this training paradigm, may hinder model performance.

## K.2 TEMPLATED LANGUAGE

Action Seq.	GPT-4o	Llama-8B-Inst	Llama-70B-Inst
1	76.92 <sub>6.75</sub>	32.50 <sub>7.40</sub>	63.41 <sub>7.52</sub>
10	66.67 <sub>7.55</sub>	19.51 <sub>6.19</sub>	47.50 <sub>7.89</sub>
19	64.10 <sub>7.68</sub>	17.95 <sub>6.15</sub>	25.64 <sub>6.99</sub>

Table 18: Performance comparison of GPT-4o, Llama-3.1-8B-Instruct, and Llama-3.1-70B-Instruct on the free-answer subset of the benchmark evaluated without the ramifications constraints using the zero-shot-CoT. The input is given in the templated language. The results are categorized up by the action sequence length.

Table 18 summarizes the performance of LLMs when inputs are presented in the templated formats described in Section 3.4 rather than in fully paraphrased natural language. The results indicate a consistent improvement across all models. Notably, GPT-4o achieves the highest gain, with an average performance improvement of 11.2%. Llama-3.1-8B-Instruct exhibits the second largest improvement, with a 6.5% increase, while Llama-3.1-70B-Instruct demonstrates a modest gain of 1.81%. These results suggest that templated inputs reduce the verbal reasoning burden on LLMs, leading to more accurate outputs.