🌍 TEXT2WORLD: Benchmarking Large Language Models for Symbolic World Model Generation

Mengkang Hu[*♠] **Tianxing Chen**[*♠] **Yude Zou**[*♣] **Yuheng Lei**[♠] **Qiguang Chen**[♡]
Ming Li[♠] **Yao Mu**[♠] **Hongyuan Zhang**[♠★] **Wenqi Shao**[♢] **Ping Luo**[†♠]
[♠] The University of Hong Kong [♣] Shenzhen University
[♡] Harbin Institute of Technology [♢] Shanghai AI Laboratory
[★] Institute of Artificial Intelligence (TeleAI), China Telecom
mkhu@connect.hku.hk, pluo.lhi@gmail.com
text-to-world.github.io

Abstract

Recently, there has been growing interest in leveraging large language models (LLMs) to generate symbolic world models from textual descriptions. Although LLMs have been extensively explored in the context of world modeling, prior studies encountered several challenges, including evaluation randomness, dependence on indirect metrics, and a limited domain scope. To address these limitations, we introduce a novel benchmark, TEXT2WORLD, based on planning domain definition language (PDDL), featuring hundreds of diverse domains and employing multi-criteria, execution-based metrics for a more robust evaluation. We benchmark current LLMs using TEXT2WORLD and find that reasoning models trained with large-scale reinforcement learning outperform others. However, even the best-performing model still demonstrates limited capabilities in world modeling. Building on these insights, we examine several promising strategies to enhance the world modeling capabilities of LLMs, including test-time scaling, agent training, and more. We hope that TEXT2WORLD can serve as a crucial resource, laying the groundwork for future research in leveraging LLMs as world models.

1 Introduction

The significance of world models for intelligent behavior has been historically acknowledged in early psychological theories, which posited that organisms employ internal representations of the external world for prediction and planning (Craik, 1967). Furthermore, LeCun (2022) extends this concept by highlighting world modeling as a core component of autonomous machine intelligence. In this paper, we primarily study *symbolic world models* (also known as domain models), which are formal representations of an environment's dynamics and constraints. In recent years, Large Language Models (LLMs) (OpenAI, 2022; Yang et al., 2024; Meta AI, 2024) have showcased their understanding of common-world knowledge, making them promising candidates for generating symbolic world models, which requires inferring action dynamics and constraints from solely natural language description (Qin et al., 2024; Chen et al., 2024b, 2025a). Some works have already explored this across numerous tasks, including planning (Hu et al., 2024b; Guan et al., 2023), game design (Wang et al., 2023a, 2024), reinforcement learning (Tang et al., 2024) among others.

Despite extensive exploration, previous work for evaluating symbolic world model generation suffers from several key limitations: *(i)* ***Limited Domain Scope***: These studies are often confined to a narrow set of domains (typically fewer than 20), which limits the generalizability and applicability of their findings (Oswald et al., 2024; Silver et al., 2024; Wong et al., 2023). *(ii)* ***Evaluation Randomness***: Some works rely on LLM-based evaluation methods, which may introduce additional margins of error (Wang et al., 2023a). Preliminary experiments in Section 3.6 demonstrate that the LLM-based evaluation exhibits a low inter-annotator agreement with human annotators (Cohen's $\kappa = 0.10$). *(iii)* ***Indirect Evaluation***: Some studies evaluate world models based on end-to-end success rates in model-based planning, making it difficult to identify specific failure modes (Guan et al., 2023; Dainese et al., 2024).

Motivated by these issues, this paper introduces a novel benchmark TEXT2WORLD based on the Planning Domain Definition Language (PDDL), as illustrated in Figure 1. Specifically, to address the first issue, we initially gathered a broad set of domains, which were then filtered through an automated pipeline and manually curated to ensure their quality, ultimately resulting in a collection of hundreds of diverse domains. Furthermore, to

[*] Equal Contribution. † Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2025, pages 26043–26066
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

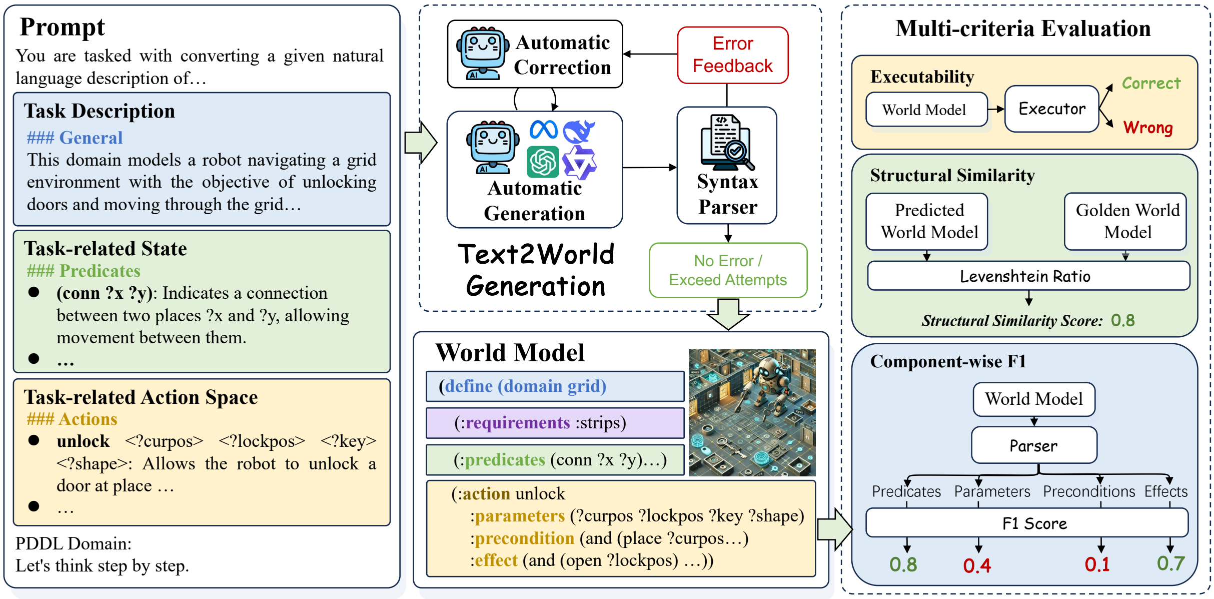

Prompt
You are tasked with converting a given natural language description of…
Task Description
### General
This domain models a robot navigating a grid environment with the objective of unlocking doors and moving through the grid…
Task-related State
### Predicates
(conn ?x ?y): Indicates a connection between two places ?x and ?y, allowing movement between them.
…
Task-related Action Space
### Actions
unlock <?curpos> <?lockpos> <?key> <?shape>: Allows the robot to unlock a door at place …
…
PDDL Domain:
Let's think step by step.
Automatic Correction
Error Feedback
Automatic Generation
Syntax Parser
Text2World Generation
No Error / Exceed Attempts
World Model
(define (domain grid)
(:requirements :strips)
(:predicates (conn ?x ?y)…)
(:action unlock
:parameters (?curpos ?lockpos ?key ?shape)
:precondition (and (place ?cupos…)
:effect (and (open ?lockpos) …))
Multi-criteria Evaluation
Executability
World Model
Executor
Correct
Wrong
Structural Similarity
Predicted World Model
Golden World Model
Levenshtein Ratio
Structural Similarity Score: 0.8
Component-wise F1
World Model
Parser
Predicates
Parameters
Preconditions
Effects
F1 Score
0.8
0.4
0.1
0.7

Figure 1: Overview of TEXT2WORLD.

tackle the second issue, we designed multi-criteria, execution-based metrics to ensure a more robust assessment. Specifically, we not only employed structural similarity for an overall evaluation but also designed component-wise F1 scores to assess finer-grained aspects such as action dynamics. Moreover, to overcome the third issue, we systematically applied these metrics to assess the generated world model directly, eliminating reliance on indirect feedback mechanisms.

We also performed data contamination analysis using n-gram matching (Touvron et al., 2023), revealing a lower contamination rate ($\mu = 0.04$) compared to prior works (Guan et al., 2023; Smirnov et al., 2024), indicating that TEXT2WORLD effectively evaluates LLMs' world modeling capabilities rather than pattern memorization.

We used TEXT2WORLD to benchmark the world modeling capabilities of 16 different LLMs from 9 model families. Experimental results in Table 1 highlight several key findings: *(i) The most advanced LLMs still struggle with* TEXT2WORLD*; (ii) large reasoning models trained by reinforcement learning show stronger world modeling capabilities;* and *(iii) error correction significantly improves model performance.* To gain a deeper understanding, we performed a manual error analysis and found that the majority were due to the LLMs' inability to include essential preconditions or effects. We also explored several strategies to enhance the world modeling capabilities of LLMs. Specifically, we initially experimented with scaling the test-time budget and observed consistent improvements as the test-time budget increased. Additionally, methods like fine-tuning and in-context learning contributed positively to model effectiveness. Moreover, we found that supervised fine-tuning on agent trajectory data yielded unexpected gains, underscoring the importance of robust world modeling for developing high-performing agents.

To facilitate further research, benchmark and code are available at this URL.

2 Preliminary

2.1 World Model

We formally define a symbolic world model as $D = \langle F, A \rangle$, where F represents the set of fluents (state variables represented as predicates) and A is the set of possible actions. Each fluent $f \in F$ is a predicate of the form $p(x_1, ..., x_n)$, where p is the predicate name and $x_1, ..., x_n$ are typed variables. Each action $a \in A$ is defined as a tuple $a = \langle \alpha, \mathcal{P}, \varphi, \mathcal{E} \rangle$ where: i) α denotes the action signature (identifier); ii) $\mathcal{P}$ represents a list of typed parameters $(p_1, ..., p_k)$; iii) φ specifies the preconditions: a logical formula over fluents that must hold for the action to be applicable; and iv) $\mathcal{E}$ defines the effects: a set of fluent literals describing how the action changes the world state.

2.2 Task Definition

The task is formally defined as: $\mathcal{M} : \mathcal{N} \to D, D \models \mathcal{N}$, where $\mathcal{M}$ is a mapping function (implemented by an LLM) that generates world model

D from the natural language description $\mathcal{N}$. $\models$ denotes semantic satisfaction. Each $\mathcal{N}$ contains the following components: i) A general description describing the overall objective of the domain; ii) A set of predicates $\mathcal{N}_F = \{f_1, ..., f_n\}$ where each predicate is described with its signature (e.g., "*(conn ?x ?y)*") and an explanation (e.g., "*Indicates a connection between two places ?x and ?y*"); iii) A set of actions $\mathcal{N}_A = \{a_1, ..., a_m\}$ where each action is described with: its signature (e.g., "*move <?curpos> <?nextpos>*") and an explanation (e.g., "*Allows the robot to move from place <?curpos> to place <?nextpos>*"). Note that to evaluate LLMs' inherent world modeling capabilities, action descriptions in $\mathcal{N}_A$ are intentionally kept at a high level, without explicit specifications of preconditions φ and effects $\mathcal{E}$. This design choice allows us to assess how well LLMs can infer the underlying world dynamics and constraints from purely descriptive text. A comparative analysis of model performance conditioned on different description styles is presented in Section 6.5.

2.3 Evaluation Metrics

We directly evaluated generated world models, addressing the ambiguity associated with indirect evaluations (Guan et al., 2023; Dainese et al., 2024). In addition, we proposed using execution-based metrics, overcoming the randomness of LLM-based evaluation (Wang et al., 2023a). Specifically, we established the following evaluation metrics: *(i)* ***Executability (Exec.)*** measures whether the generated PDDL can be successfully parsed and validated by standard PDDL validators. *(ii)* ***Structural Similarity (Sim.)*** quantifies the textual similarity between the generated and ground truth PDDL using normalized Levenshtein ratio. *(iii)* ***Component-wise F1 Scores:*** When generated PDDL achieves executability (Exec. = 1), we perform fine-grained analysis by calculating the macro-averaged F1 score for each component type (predicates, actions, etc.). More specifically, we compute F1 scores for predicates ($\mathbf{F1_{PRED}}$), parameters ($\mathbf{F1_{PARAM}}$), preconditions ($\mathbf{F1_{PRECOND}}$), and effects ($\mathbf{F1_{EFF}}$) by parsing both generated and ground truth PDDL into structured representations.

3 Benchmark Construction

The overall process of benchmark construction is shown in Figure 2. In this section, we provide a detailed explanation of each stage.

3.1 Data Acquisition

Our benchmark construction process began with collecting PDDL files from various public repositories and planning competitions. Through this initial collection phase, we accumulated 1,801 raw PDDL files. We performed several preprocessing steps to standardize the data format (e.g., convert files with BOM encoding to standard UTF-8). The processed files served as the foundation for our dataset construction.

3.2 Data Filtering and Manual Selection

To ensure the quality and reliability of Text2World, we implemented a comprehensive filtering pipeline: *(i)* ***Validation:*** We employed a PDDL domain parser to perform syntax validation on each file; *(ii)* ***Similarity Deduplication:*** We eliminated duplicate entries by computing pairwise cosine similarity on TF-IDF vectorized PDDL content, removing files with similarity scores exceeding 0.9; *(iii)* ***Complexity Control:*** We removed domains with over 40 predicates or 20 actions to balance expressiveness with practical utility. *(iv)* ***Token Length Filtering:*** We removed files exceeding 5,000 tokens using GPT-2 (Radford et al., 2019) tokenizer to ensure compatibility with model context windows. Additionally, we conducted manual selection to eliminate domains that were not designed for world modeling (such as blocksworld-mystery) and low-quality cases that were not captured by the automated filtering methods. After this process, we obtained 264 high-quality PDDL domain specifications.

3.3 Data Annotation

After obtaining the high-quality PDDL domains, we manually annotated natural language descriptions for each domain. To ensure the quality of annotations, we recruited 6 computer science graduates as annotators. The annotated description followed the structured format described in Section 2.2, and annotators were required to follow the annotation criteria: *(i)* ***Descriptive Completeness:*** Annotations must contain all required components; *(ii)* ***Action Abstraction:*** Action descriptions should avoid explicit references to formal preconditions and effects; *(iii)* ***Inference-Enabling:*** Descriptions should contain sufficient contextual information to allow models to infer the underlying dynamics; *(iv)* ***Natural Language Priority:*** Technical terminol-

TSP Domain Example

```
(define (domain tsp)
    (:predicates
        (at ?x)
        (visited ?x))

    (:action move
        :parameters (?x ?y)
        :precondition (at ?x)
        :effect (and (at ?y)
                (visited ?y)
                (not (at ?x)))))
```

Abstract Description

General.
This domain models a simplified version of the Traveling Salesperson Problem (TSP), where the goal is to visit a series of locations. The domain includes actions for moving between locations and marking locations as visited.

Predicates.

- `(at ?x)`: Indicates the current location.
- `(lock-shape ?x ?s)`: Indicates that location ?x has been visited.

Actions.

- `move <?x> <?y>`: Allows moving from location ?x to location ?y.

Concrete Description

General.
This domain models a simplified version of the Traveling Salesperson Problem (TSP), where the goal is to visit a series of locations. The domain includes actions for moving between locations and marking locations as visited.

Predicates.

- `(at ?x)`: Indicates the current location.
- `(lock-shape ?x ?s)`: Indicates that location ?x has been visited.

Actions.

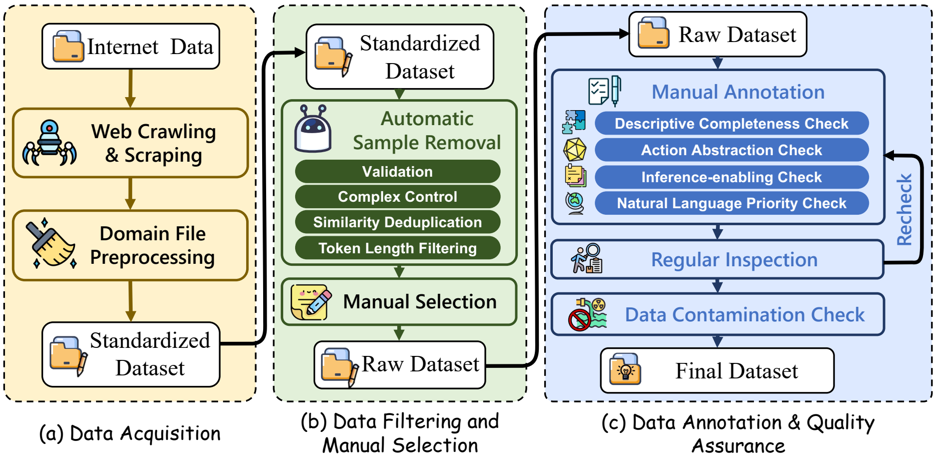
- `move <?x> <?y>`: Allows moving from location ?x to location ?y. The precondition for this action is being at location ?x. The effect of this action is that the agent is now at location ?y, location ?y is marked as visited, and the agent is no longer at location ?x.

ogy should be minimized in favor of natural language explanations. Examples of TEXT2WORLD can be found in Appendix A.1.

3.4 Quality Assurance

Manual Recheck To maintain rigorous quality standards throughout the annotation process, we established a review system supervised by two senior experts. These experts conducted regular inspections of the annotations, ensuring accuracy and consistency. Inspectors must verify all data twice to determine if the annotated examples meet the specified annotation standards. Examples are accepted only if both inspectors approve them. The verification results showed "almost perfect agreement" with a Fleiss Kappa (Landis and Koch, 1977) score of 0.82. Through this comprehensive quality control process, we compiled a final curated dataset of 103 domains with gold-standard descriptions.

Data Contamination As shown by Carlini et al. (2021), LLMs can memorize training data rather than truly model the world. To assess potential contamination between LLMs' training data and TEXT2WORLD, we generated complete PDDL domains from the first 20 tokens using GPT-4 (OpenAI, 2023) and calculated contamination rates based on tokenized 10-grams with up to 4 mismatches (Touvron et al., 2023), excluding PDDL-specific keywords and variables. We also compared these results with previous studies (Guan et al., 2023; Smirnov et al., 2024). Figure 3 shows that TEXT2WORLD has a lower contamination rate ($\mu = 0.04$ vs. $\mu = 0.47$), suggesting its performance reflects domain understanding rather than memorization. However, the complete elimination of contamination remains challenging due to PDDL's widespread use.



Statistic	Number
Domain Count	
Domain	103
- Train	2
- Test	101
Token Count	
Description	851.6 ± 515.2
- Min/Max	[159, 2814]
Domain	1187.2 ± 1212.1
- Min/Max	[85, 7514]
Line Count	
Domain	75.4 ± 62.9
- Min/Max	[9, 394]
Component Count	
Actions	4.5 ± 2.8
- Min/Max	[1, 16]
Predicates	8.1 ± 4.8
- Min/Max	[1, 25]
Types	1.1 ± 1.3
- Min/Max	[1, 8]

Figure 2: *Left*: Dataset construction process including: (a) *Data Acquisition* (§3.1); (b) *Data Filtering and Manual Selection* (§3.2); (c) *Data Annotation and Quality Assurance* (§3.3 and §3.4). *Right*: Key statistics of `Text2World`. Tokens are counted by GPT-2 (Radford et al., 2019) tokenizer. The style is referenced from Chen et al. (2024c).

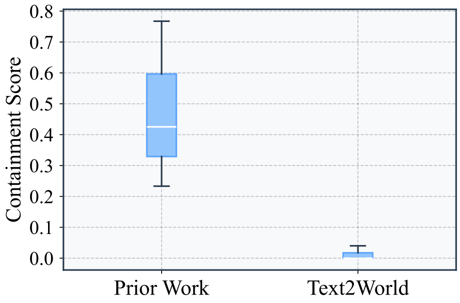


Figure 3: n-gram contamination rate of TEXT2WORLD and prior works.

3.5 Data Analysis

This section provides some detailed data analysis to better understand TEXT2WORLD.

Core Statistics We designated 2 domains as in-context exemplars (train set), with the remaining 101 samples forming our test set.

Semantic Analysis We use LLMs to extract high-level domain characteristics to better understand the conceptual distribution of TEXT2WORLD, As shown in Figure 4 (Bottom), common themes such as *path planning*, *constraint satisfaction*, and *task allocation*, among others, emerge.

Requirements Analysis A PDDL requirement specifies a formal capability needed to express a domain, often reflecting its complexity. For instance, `:typing` stands for allowing the usage of typing for objects. As shown in Figure 4 (Top), there are eight different requirement type in TEXT2WORLD. We also provide an in-depth analysis of requirement type in Appendix A.3.

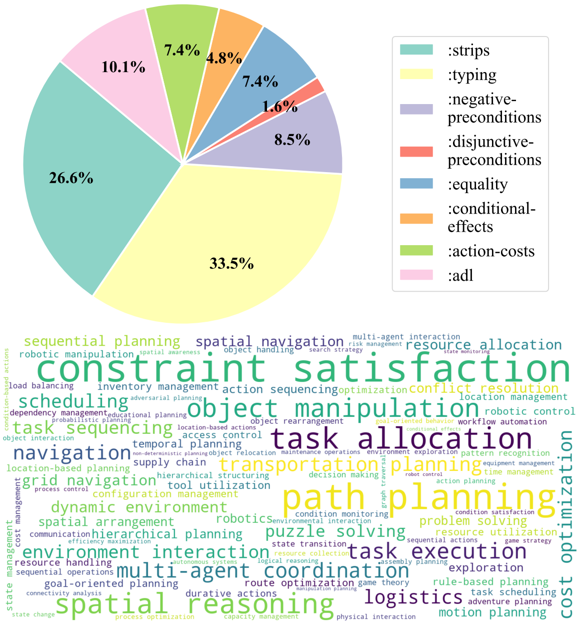


Figure 4: *Top*: The frequency of requirements distribution. *Bottom*: Word cloud of concepts in TEXT2WORLD.

3.6 Preliminary Experiment

In previous works, LLMs have been employed to evaluate the action dynamics of world models generated by LLMs themselves (Wang et al., 2023a). To further assess the ability of LLMs to detect errors in world models, we conducted a preliminary experiment where we first used `claude-3.5-sonnect` for TEXT2WORLD. Subsequently, human annotators and the LLM independently evaluated the generated action dynamics to identify potential errors. The inter-annotator agreement between human ratings and LLM ratings, measured using Cohen's κ, was 0.10, indicating a low level of agreement. This suggests that predicting the correctness of PDDL domains using an LLM is particularly challenging, highlighting the need for more discriminative evaluation metrics. Prompting examples and more results can be found in Appendix A.2.

4 Experiments

4.1 Experimental Setup

We evaluate several state-of-the-art LLMs, including *GPT-4* (OpenAI, 2023), *GPT-3.5* (OpenAI, 2022), *Claude-3.5* (Anthropic), and *LLaMA-3.1* (Meta AI), *DeepSeek-v3* (Liu et al., 2024), *CodeLlaMA* (Roziere et al., 2023), *LlaMA-2* (Touvron et al., 2023), etc. We also evaluated Large Reasoning Models (LRMs) trained using reinforcement learning, such as *DeepSeek-R1* (DeepSeek-AI et al., 2025), *OpenAI-o1* (OpenAI, 2024) and *OpenAI-o3* (OpenAI, 2025). We set temperature = 0 for each model for all experiments to maintain reproducibility. We employ tarski [1] library to check syntactic correctness and executability. We prompt LLMs to generate symbolic world models under a zero-shot setting with chain-of-thought reasoning (Wei et al., 2022). In error-correction experiments, LLMs refine outputs based on validator-reported syntax errors, denoted as EC_3 for k attempts. Evaluation of open-sourced models were conducted on NVIDIA A100 GPUs with 80GB memory. We access proprietary models through their official API platform. Prompt examples can be found in Appendix B.2.

4.2 Experimental Results

Several conclusions can be drawn from Table 1: *(i)* ***The most advanced LLMs still struggle with*** **TEXT2WORLD.** For example, the best-performing model, *DeepSeek-R1*, achieves F1 scores below 60% for both preconditions ($\text{F1}_{\text{PRECOND}}$) and effects ($\text{F1}_{\text{EFF}}$) under the without error correction setting. This highlights the limitations of current LLMs in world modeling tasks. *(ii)* ***Large reasoning models trained with reinforcement learning exhibit superior world modeling capabilities.*** These models, such as *DeepSeek-R1* (DeepSeek-AI et al., 2025), outperform others in executability, structural similarity, and component-wise performance, indicating that RL-based training enhances the ability of models to generate structured and valid world models. *(iii)* ***The ability of models to benefit from error correction is evident.*** For instance, *GPT-4* (`gpt-4o-mini`) demonstrates a notable improvement in executability, increasing from 48.5% to 72.3% after three correction attempts.

5 Analysis

5.1 Statistical Analysis

We conducted a one-way ANOVA (Girden, 1992) to evaluate the impact of correction attempts on model performance, excluding anomalous zero values. The results showed a significant improvement with three correction attempts ($F = 27.48, p = 0.00012$), indicating that correction attempts lead to a notable enhancement in model performance.

5.2 Error Analysis

The interpretable nature of generating symbolic world models can be utilized for a deeper manual analysis of the failure modes. We select the results from `claude-3.5-sonnect` under the few-shot setting for manual error analysis. Errors are categorized into syntax and semantic errors, where syntax errors occur when the generated domain cannot be validated ($\text{EXEC.} = 0$), and semantic errors arise when the generated world model does not align with action dynamics or fails to follow the natural language description. More details and the error analysis for `deepseek-r1` are presented in Appendix C.

Syntax Errors Figure 5 (*Left*) shows the distribution of syntax errors during correction. Common errors like `UndefinedConstant` and `IncorrectParentheses` decrease over correction steps, indicating improvements in syntax validation, though errors like `UndefinedDomainName` and `UndefinedType` persist.

Semantic Error Figure 5 (*Right*) illustrates the distribution of semantic errors. Semantic errors are categorized into four types: *(i)* `DisobeyDescription` involves direct violations of descriptions. *(ii)* `IncompleteModeling`, where the world model lacks necessary components. *(iii)* `RedundantSpecifications` refers to superfluous preconditions or effects; and *(iv)* `SurfaceDivergence` involves surface-level variations that preserve semantic equivalence to gold domain. In addition, since a domain may encompass various action dynamics, different error types can occur simultaneously. For instance, nearly 10%

[1] https://github.com/aig-upf/tarski

Table 1: Performance comparison of different LLMs on TEXT2WORLD. EC_k denotes the setting where models are allowed k correction attempts (EC_0: zero-shot without correction, EC_3: with 3 correction attempts).

Model Family	**Version**	**EXEC.** ↑		**SIM.** ↑		$\mathbf{F1_{PRED}}$ ↑		$\mathbf{F1_{PARAM}}$ ↑		$\mathbf{F1_{PRECOND}}$ ↑		$\mathbf{F1_{EFF}}$ ↑	
		$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$
OPENAI-O1	`o1-mini`	49.5	69.3	82.5	82.2	48.4	66.3	36.4	49.7	28.9	38.0	31.7	42.1
OPENAI-O3	`o3-mini`	54.5	84.2	83.0	81.9	53.9	81.1	43.7	63.0	36.8	50.4	39.4	53.8
GPT-4	`gpt-4o`	60.4	75.2	84.5	84.1	59.6	72.1	56.5	68.1	49.3	56.4	47.8	56.7
	`gpt-4o-mini`	48.5	72.3	82.6	82.2	48.1	70.1	47.1	67.3	34.9	47.5	38.2	52.7
GPT-3.5	`turbo-0125`	41.6	56.4	81.9	81.6	41.2	55.8	39.6	53.8	30.2	39.2	27.5	37.7
CLAUDE-3.5	`sonnet`	45.5	64.4	73.2	66.8	45.5	62.5	41.5	48.8	37.4	44.0	38.4	45.0
LLAMA-2	`7b-instruct`	0.0	0.0	45.5	33.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	`70b-instruct`	0.0	0.0	48.7	48.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LLAMA-3.1	`8b-instruct`	0.0	0.0	74.3	74.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	`70b-instruct`	0.0	0.0	83.6	79.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DEEPSEEK	`deepseek-v3`	56.4	79.2	**84.7**	**84.2**	55.9	75.6	53.7	74.4	45.1	58.6	46.7	61.5
	`deepseek-r1`	**72.3**	**89.1**	84.3	84.0	**71.7**	**86.7**	**64.0**	**76.3**	**57.6**	**65.0**	**58.8**	**67.3**
CODELLAMA	`7b-instruct`	17.8	22.8	60.2	57.6	17.8	18.8	17.2	18.2	11.3	12.2	10.7	11.1
	`13b-instruct`	7.9	8.9	57.6	55.0	7.9	8.9	7.9	8.9	4.9	5.9	5.2	6.1
	`34b-instruct`	7.9	8.9	34.2	7.6	7.9	8.6	7.9	8.4	5.0	5.0	5.4	5.4
	`70b-instruct`	16.8	16.8	54.0	14.0	16.4	16.4	16.8	16.8	10.7	10.7	14.1	14.1

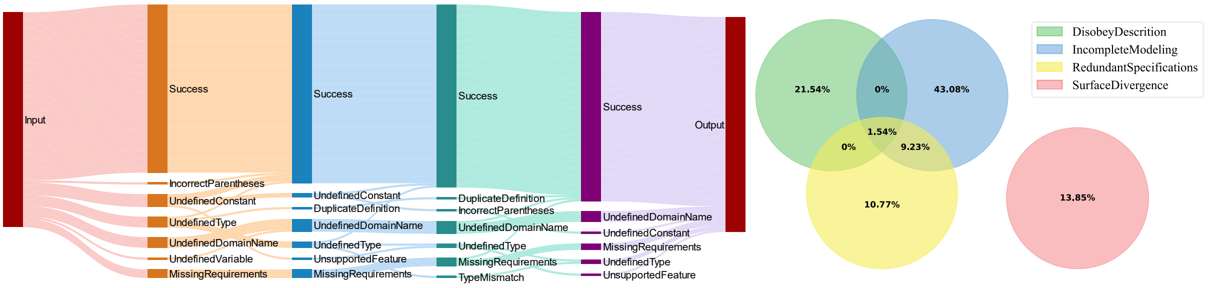


Figure 5: *Left*: The distribution of syntax error types for `claude-3.5-sonnect` during the progression of correction. *Right*: The distribution of semantic error types.

of cases exhibited both `IncompleteModeling` and `RedundantSpecifications` concurrently.

6 Exploration

In addition to the zero-shot CoT evaluation in Section 4.2, we further evaluate the models on TEXT2WORLD with five different strategies: (1) *Test-time Scaling*; (2) *In-Context Learning*; (3) *Fine-tuning*; (4) *Agent Training*; (5) *Inference with Concrete Description*.

6.1 Test-time Scaling

Recently, test-time scaling has demonstrated remarkable potential (OpenAI, 2024; DeepSeek-AI et al., 2025). We use the error information from the syntax parser as feedback and assess whether increasing the test-time compute budget can enhance the LLM's performance. As shown in Figure 6, the model exhibits consistent improvement with increased test-time computation. More advanced test-time scaling strategies may serve as a viable approach to enhancing the model's world modeling ability (Chen et al., 2025b).

6.2 In-Context Learning

We also perform a few-shot evaluation in Section 6.2, where we carefully select demonstration "*gripper*" and "*blocks*" that are structurally similar but semantically distinct from the test cases to prevent data leakage. As shown in Table 2, we observe that different models exhibit varying degrees of improvement from in-context learning. For instance, `claude-3.5-sonnect` demonstrates a substantial enhancement, achieving over a 20%

Table 2: The experimental results of models under different settings: (1) In-context learning (§6.2); (2) Fine-tuning, and fine-tuning with LoRA (Hu et al., 2021) (§6.3); (3) Agent training (§6.4).

Model Family	**EXEC.**		**SIM.**		**F1$_{\text{PRED}}$**		**F1$_{\text{PARAM}}$**		**F1$_{\text{PRECOND}}$**		**F1$_{\text{EFF}}$**	
	EC$_0$	**EC$_3$**	**EC$_0$**	**EC$_3$**	**EC$_0$**	**EC$_3$**	**EC$_0$**	**EC$_3$**	**EC$_0$**	**EC$_3$**	**EC$_0$**	**EC$_3$**
In-Context Learning												
CLAUDE-3.5-SONNET	45.5	64.4	73.2	66.8	45.5	62.5	41.5	48.8	37.4	44.0	38.4	45.0
w. 2-SHOT	78.2$_{+32.7}$	88.1$_{+23.7}$	83.9$_{+10.7}$	82.3$_{+15.5}$	77.0$_{+31.5}$	86.1$_{+23.6}$	75.2$_{+33.7}$	82.1$_{+33.3}$	65.6$_{+28.2}$	71.3$_{+27.3}$	67.2$_{+28.8}$	73.4$_{+28.4}$
DEEPSEEK-R1	72.3	89.1	84.3	84.0	71.7	86.7	64.0	76.3	57.6	65.0	58.8	67.3
w. 2-SHOT	69.3$_{-3.0}$	90.1$_{+1.0}$	83.8$_{-0.5}$	83.5$_{-0.5}$	68.4$_{-3.3}$	87.7$_{+1.0}$	64.6$_{+0.6}$	79.1$_{+2.8}$	56.0$_{-1.6}$	66.9$_{+1.9}$	57.6$_{-1.2}$	68.9$_{+1.6}$
GPT-4O-MINI	48.5	72.3	82.6	82.2	48.1	70.1	47.1	67.3	34.9	47.5	38.2	52.7
w. 2-SHOT	40.6$_{-7.9}$	69.3$_{-3}$	82.9$_{+0.3}$	82.4$_{+0.2}$	40.3$_{-7.8}$	67.2$_{-2.9}$	40.1$_{-7}$	67.0$_{-0.3}$	31.6$_{-3.3}$	49.3$_{+1.8}$	32.5$_{-5.7}$	54.8$_{+2.1}$
Fine-tuning (FT)												
LLAMA-3.1-8B	0.0	0.0	74.3	74.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w. FT	52.5$_{+52.5}$	68.3$_{+68.3}$	80.8$_{+6.5}$	80.6$_{+5.7}$	51.4$_{+51.4}$	65.4$_{+65.4}$	48.5$_{+48.5}$	60.6$_{+60.6}$	31.5$_{+31.5}$	38.1$_{+38.1}$	32.4$_{+32.4}$	40.2$_{+40.2}$
LLAMA-3.1-70B	0.0	0.0	83.6	79.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w. LORA	48.5$_{+48.5}$	70.3$_{+70.3}$	83.8$_{+0.2}$	82.3$_{+3.1}$	47.9$_{+47.9}$	68.5$_{+68.5}$	48.5$_{+48.5}$	66.4$_{+66.4}$	39.9$_{+39.9}$	52.8$_{+52.8}$	40.6$_{+40.6}$	52.1$_{+52.1}$
Agent Training (AT)												
LLAMA-2-70B	0.0	0.0	48.7	48.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w. AT	7.9$_{+7.9}$	9.9$_{+9.9}$	65.6$_{+16.9}$	47.9$_{-0.7}$	7.3$_{+7.3}$	8.8$_{+8.8}$	7.3$_{+7.3}$	9.1$_{+9.1}$	6.1$_{+6.1}$	6.5$_{+6.5}$	5.7$_{+5.7}$	6.1$_{+6.1}$

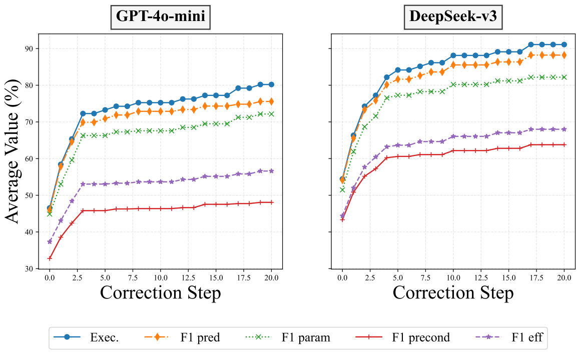


Figure 6: The performance of `gpt-4o-mini` (left) and `deepseek-v3` (right) under different test-time compute budgets, showing consistent improvement with increased compute.

increase in the component-wise F1 score. However, for `gpt-4o-mini`, incorporating few-shot examples resulted in a decrease in model performance.

6.3 Fine-tuning

We leverage the AgentGen (Hu et al., 2024b) framework to synthesize 601 PDDL domains and their corresponding descriptions for fine-tuning *LLaMA-3.1* (Meta AI) to investigate potential improvements in their world modeling capabilities. As shown in Table 2, fine-tuning can lead to significant improvements in model performance. For instance, the fine-tuned Llama-3.1-70B demonstrated performance comparable to GPT-4o-mini, highlighting that supervised fine-tuning is an effective method for bridging the gap between open-source and proprietary models. Moreover, larger models tend to benefit more from supervised fine-tuning, with the 70B LLaMA-3.1 showing greater improvement than the 8B model.

6.4 Agent Training

Many studies have demonstrated that supervised fine-tuning on agent trajectories can enhance a model's performance on agentic tasks (Hu et al., 2024b; Zeng et al., 2023) (i.e., agent training). Some previous works also discussed that a good agent model requires a sufficiently strong internal world representation (LeCun, 2022). Therefore, in this section, we explore whether agent training can improve the model's world modeling capabilities. More specifically, we trained *LLaMA-2-70B* model on AgentInstruct (Zeng et al., 2023). As shown in Table 2, the model's world modeling capabilities are enhanced post-agent training, indicating a positive correlation between performance on agentic tasks and the model's world modeling abilities.

6.5 Inference with Concrete Description

As is discussed in Section 2.2, we intentionally make the natural language description of a world model at a high level. We refer to these high-level descriptions as "*abstract descriptions*," in contrast to more detailed "*concrete descriptions*" that explicitly specify preconditions and effects. Examples of both description types can be found in the Appendix A.1.2. Using concrete descriptions simplifies the task by requiring the model to directly map the provided text to a world specification, bypassing the need to infer symbolic action dynamics. The observed consistent improvement (as shown in Figure 7) supports the claim that the model's

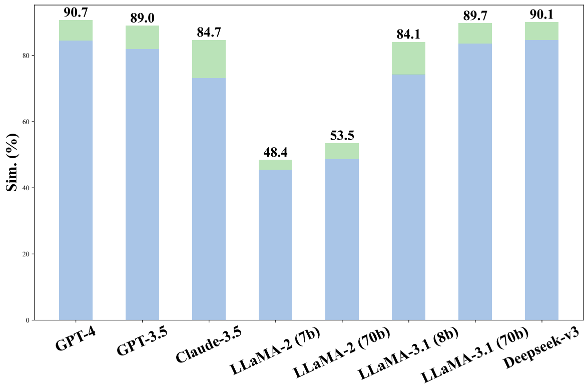


Figure 7: Comparison of model performance on abstract versus concrete domain descriptions, showing the base score for abstract descriptions (blue) and the improvement gained from concrete descriptions (green).

ability to deduce action dynamics from abstract descriptions is still lacking. We also provide more detailed experimental results in Appendix D.1.

7 Related Work

Neural world modeling is a long-standing research topic with widespread applications across various fields, including reinforcement learning (Ha and Schmidhuber, 2018b,a), robotics (Wu et al., 2023), and autonomous driving (Guan et al., 2024), among others. In recent years, LLMs trained on massive datasets have demonstrated zero-shot capabilities across a variety of tasks, including planning (Zhao et al., 2023; Qin et al., 2024; Huang et al., 2022; Hu et al., 2024a), robotics (Mu et al., 2024; Chen et al., 2024a), analog design (Lai et al., 2024), and more. Preliminary studies propose directly using LLMs as world models (Hao et al., 2023; Wang et al., 2024, 2023b; Li et al., 2022), by taking the state and action as input and predicting the next state, but the unreliability and limited interpretability of LLM outputs can lead to accumulating errors. Moreover, some studies have shown that autoregressive models perform poorly in predicting action effects (Banerjee et al., 2020; Luo et al., 2023). Tree-planner (Hu et al., 2023) instead proposes to constructing the possible action space using LLMs before executing. Another line of work focuses on leveraging LLMs to construct symbolic world models (Oswald et al., 2024; Silver et al., 2024; Smirnov et al., 2024; Zhu et al., 2024; Wang et al., 2023a; Wong et al., 2023; Vafa et al., 2024). For example, Guan et al. (2023) uses LLMs to generate a PDDL domain model and relies on human feedback to correct errors. AgentGen (Hu et al., 2024b) synthesizes diverse PDDL domains, aiming to create high-quality planning data. Xie et al. (2024) propose to finetune LLMs for predicting precondition and effect of actions. Despite the growing interest in this research direction, there is currently a lack of a comprehensive benchmark in this area.

8 Conclusion

We present TEXT2WORLD, a novel benchmark consisting of hundreds of domains designed to evaluate the world modeling capabilities of large language models (LLMs). Developed through a meticulous and thorough process, TEXT2WORLD provides a robust foundation for analysis. Additionally, we conducted an extensive evaluation involving 16 different LLMs from 9 model families based on TEXT2WORLD. We hope that TEXT2WORLD will inspire future research in leveraging LLMs as world models.

Ethical Considerations

Data Access. We collected data from open-source repositories and ensured that these repositories are available for academic research in accordance with our commitment to ethical data use.

Participant Recruitment. We recruited graduate students as annotators and required all participants to achieve an IELTS score of 6 or above. To mitigate potential biases stemming from participants' geographical backgrounds, we minimized national differences in the dataset by focusing on human commonsense. All annotators provided informed consent and were compensated above the local minimum wage—$10 per hour for standard annotators and $20 per hour for senior annotators.

Potential Risk. We confirmed that our dataset does not contain any personal data (e.g., names, contacting information), and our data collection procedures adhere to ethical guidelines.

Limitation

Due to the limited number of available domains online, we did not construct a large-scale training set. Future work should focus on expanding the dataset by incorporating additional data sources, such as synthesized data (Hu et al., 2024b), to cover a broader range of domains. Furthermore, although we conducted regular inspections to minimize the introduction of subjectivity into the dataset, the unavoidable influence of human subjectivity during manual annotation may introduce potential biases.

References

Anthropic. Introducing claude 3.5 sonnet.

Pratyay Banerjee, Chitta Baral, Man Luo, Arindam Mitra, Kuntal Pal, Tran C Son, and Neeraj Varshney. 2020. Can transformers reason about effects of actions? *arXiv preprint arXiv:2012.09938*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Junting Chen, Checheng Yu, Xunzhe Zhou, Tianqi Xu, Yao Mu, Mengkang Hu, Wenqi Shao, Yikai Wang, Guohao Li, and Lin Shao. 2024a. **EMOS**: **E**mbodiment-aware heterogeneous **M**ulti-robot **O**perating **S**ystem with llm agents. *Preprint*, arXiv:2410.22662.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiaqi Wang, Mengkang Hu, Zhi Chen, Wanxiang Che, and Ting Liu. 2025b. Ecm: A unified electronic circuit model for explaining the emergence of in-context learning and chain-of-thought in large language model. *arXiv preprint arXiv:2502.03325*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024b. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904.

Qiguang Chen, Libo Qin, Jin Zhang, Zhi Chen, Xiao Xu, and Wanxiang Che. 2024c. M^{3}CoT: A novel benchmark for multi-domain multi-step multi-modal chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8199–8221, Bangkok, Thailand. Association for Computational Linguistics.

Kenneth James Williams Craik. 1967. *The nature of explanation*, volume 445. CUP Archive.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. 2024. Generating code world models with large language models guided by monte carlo tree search. *arXiv preprint arXiv:2405.15383*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ellen R Girden. 1992. *ANOVA: Repeated measures*. 84. Sage.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094.

Yanchen Guan, Haicheng Liao, Zhenning Li, Jia Hu, Runze Yuan, Yunjian Li, Guohui Zhang, and Chengzhong Xu. 2024. World models for autonomous driving: An initial survey. *IEEE Transactions on Intelligent Vehicles*.

David Ha and Jürgen Schmidhuber. 2018a. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31.

David Ha and Jürgen Schmidhuber. 2018b. World models. *arXiv preprint arXiv:1803.10122*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2024a. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. *arXiv preprint arXiv:2408.09559*.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. 2023. Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, Saravan Rajmohan, and Dongmei Zhang. 2024b. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.

Yao Lai, Sungyoung Lee, Guojin Chen, Souradip Poddar, Mengkang Hu, David Z Pan, and Ping Luo. 2024. Analogcoder: Analog circuit design via training-free code generation. *arXiv preprint arXiv:2405.14918*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Yann LeCun. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2022. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Man Luo, Shrinidhi Kumbhar, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, Chitta Baral, et al. 2023. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models. *arXiv preprint arXiv:2310.00836*.

Meta AI. Introducing llama 3.1: Our most capable models to date.

Meta AI. 2024. Introducing meta Llama 3: The most capable openly available LLM to date. Accessed: 2024-04-18.

Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. 2024. Robocodex: Multimodal code generation for robotic behavior synthesis. *arXiv preprint arXiv:2402.16117*.

OpenAI. 2022. Openai: Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. Learning to reason with llms.

OpenAI. 2025. Openai o3-mini.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. 2024. Large language models as planning domain generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 423–431.

Libo Qin, Qiguang Chen, Xiachong Feng, Yang Wu, Yongheng Zhang, Yinghui Li, Min Li, Wanxiang Che, and Philip S Yu. 2024. Large language models meet nlp: A survey. *arXiv preprint arXiv:2405.12819*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20256–20264.

Pavel Smirnov, Frank Joublin, Antonello Ceravola, and Michael Gienger. 2024. Generating consistent pddl domains with large language models. *arXiv preprint arXiv:2404.07751*.

Hao Tang, Darren Key, and Kevin Ellis. 2024. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. *arXiv preprint arXiv:2402.12275*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Keyon Vafa, Justin Y Chen, Jon Kleinberg, Sendhil Mullainathan, and Ashesh Rambachan. 2024. Evaluating the world model implicit in a generative model. *arXiv preprint arXiv:2406.03689*.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. 2024. Can language models serve as text-based world simulators? *arXiv preprint arXiv:2406.06485*.

Ruoyao Wang, Graham Todd, Eric Yuan, Ziang Xiao, Marc-Alexandre Côté, and Peter Jansen. 2023a. Bytesized32: A corpus and challenge task for generating task-specific world models expressed as text games. *arXiv preprint arXiv:2305.14879*.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Lionel Wong, Gabriel Grand, Alexander K Lew, Noah D Goodman, Vikash K Mansinghka, Jacob Andreas, and Joshua B Tenenbaum. 2023. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. 2023. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR.

Kaige Xie, Ian Yang, John Gunerli, and Mark Riedl. 2024. Making large language models into world models with precondition and effect knowledge. *arXiv preprint arXiv:2409.12278*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Wang Zhu, Ishika Singh, Robin Jia, and Jesse Thomason. 2024. Language models can infer action semantics for classical planners from environment feedback. *arXiv preprint arXiv:2406.02791*.

A Benchmark Construction

A.1 Example

A.1.1 Domain Example

```
(define (domain grid)
  (:requirements :strips)
  (:predicates (conn ?x ?y) (key-shape ?k ?s) (lock-shape ?x ?s)
         (at ?r ?x ) (at-robot ?x) (place ?p) (key ?k) (shape ?s)
         (locked ?x) (holding ?k)  (open ?x)  (arm-empty ))

  (:action unlock
    :parameters (?curpos ?lockpos ?key ?shape)
    :precondition (and (place ?curpos) (place ?lockpos) (key ?key)
           (shape ?shape) (conn ?curpos ?lockpos)
           (key-shape ?key ?shape) (lock-shape ?lockpos ?shape)
           (at-robot ?curpos) (locked ?lockpos) (holding ?key))
    :effect (and (open ?lockpos) (not (locked ?lockpos))))

  (:action move
    :parameters (?curpos ?nextpos)
    :precondition (and (place ?curpos) (place ?nextpos) (at-robot ?curpos)
           (conn ?curpos ?nextpos) (open ?nextpos))
    :effect (and (at-robot ?nextpos) (not (at-robot ?curpos))))

  (:action pickup
    :parameters (?curpos ?key)
    :precondition (and (place ?curpos) (key ?key) (at-robot ?curpos)
           (at ?key ?curpos) (arm-empty ))
    :effect (and (holding ?key) (not (at ?key ?curpos)) (not (arm-empty ))))

  (:action pickup-and-loose
    :parameters (?curpos ?newkey ?oldkey)
    :precondition (and (place ?curpos) (key ?newkey) (key ?oldkey)
           (at-robot ?curpos) (holding ?oldkey)
           (at ?newkey ?curpos))
    :effect (and (holding ?newkey) (at ?oldkey ?curpos)
     (not (holding ?oldkey)) (not (at ?newkey ?curpos))))

  (:action putdown
    :parameters (?curpos ?key)
    :precondition (and (place ?curpos) (key ?key) (at-robot ?curpos)
           (holding ?key))
    :effect (and (arm-empty ) (at ?key ?curpos) (not (holding ?key))))
  )
```

Listing 1: Grid PDDL

A.1.2 Abstract Description

General. This domain models a robot navigating a grid environment with the objective of unlocking doors and moving through the grid. The robot can carry keys that match the shape of locks to unlock doors. The environment includes places, keys with specific shapes, and doors (locks) with corresponding shapes that need to be unlocked.

Predicates. The following predicates are used in the domain:

- `(conn ?x ?y)`: Indicates a connection between two places ?x and ?y, allowing movement between them.
- `(key-shape ?k ?s)`: Indicates that key ?k has shape ?s.
- `(lock-shape ?x ?s)`: Indicates that lock (or door) at place ?x has shape ?s.
- `(at ?r ?x)`: Indicates that key `?r` is at place `?x`.
- `(at-robot ?x)`: Indicates that the robot is at place `?x`.

- `(place ?p)`: Indicates that `?p` is a place in the grid.
- `(key ?k)`: Indicates that `?k` is a key.
- `(shape ?s)`: Indicates that `?s` is a shape.
- `(locked ?x)`: Indicates that the place `?x` is locked.
- `(holding ?k)`: Indicates that the robot is holding key `?k`.
- `(open ?x)`: Indicates that the place `?x` is open.
- `(arm-empty)`: Indicates that the robot's arm is empty.

Actions. The following actions are available in the domain:

- `unlock <?curpos> <?lockpos> <?key> <?shape>`: Allows the robot to unlock a door at place `<?lockpos>` using a key of a specific shape.
- `move <?curpos> <?nextpos>`: Allows the robot to move from place `<?curpos>` to place `<?nextpos>`.
- `pickup <?curpos> <?key>`: Allows the robot to pick up a key at its current location.
- `pickup-and-loose <?curpos> <?newkey> <?oldkey>`: Allows the robot to pick up a new key while dropping the one it was holding.
- `putdown <?curpos> <?key>`: Allows the robot to put down a key it is holding.

A.1.3 Concrete Description

General. This domain models a robot navigating a grid environment with the objective of unlocking doors and moving through the grid. The robot can carry keys that match the shape of locks to unlock doors. The environment includes places, keys with specific shapes, and doors (locks) with corresponding shapes that need to be unlocked.

Predicates. The following predicates are used in the domain:

- `(conn ?x ?y)`: Indicates a connection between two places `?x` and `?y`, allowing movement between them.
- `(key-shape ?k ?s)`: Indicates that key `?k` has shape `?s`.
- `(lock-shape ?x ?s)`: Indicates that lock (or door) at place `?x` has shape `?s`.
- `(at ?r ?x)`: Indicates that key `?r` is at place `?x`.
- `(at-robot ?x)`: Indicates that the robot is at place `?x`.
- `(place ?p)`: Indicates that `?p` is a place in the grid.
- `(key ?k)`: Indicates that `?k` is a key.
- `(shape ?s)`: Indicates that `?s` is a shape.
- `(locked ?x)`: Indicates that the place `?x` is locked.
- `(holding ?k)`: Indicates that the robot is holding key `?k`.
- `(open ?x)`: Indicates that the place `?x` is open.
- `(arm-empty)`: Indicates that the robot's arm is empty.

Actions. The following actions are available in the domain:

- unlock <?curpos> <?lockpos> <?key> <?shape>: Allows the robot to unlock a door at place <?lockpos> using a key of a specific shape if the robot is at place <?curpos>, the key matches the lock's shape, the robot is holding the key, there is a connection between <?curpos> and <?lockpos>, and the destination is locked. After the action, the lock is no longer locked.

- move <?curpos> <?nextpos>: Allows the robot to move from place <?curpos> to place <?nextpos> if there is a connection between them and the destination is open. After the move, the robot is no longer at the original place.

- pickup <?curpos> <?key>: Allows the robot to pick up a key at its current location if the robot's arm is empty and it is at the same place as the new key. After the action, the robot is holding the key, and the key is no longer at that location.

- pickup-and-loose <?curpos> <?newkey> <?oldkey>: Allows the robot to pick up a new key while dropping the one it was holding if it is at the same place as the new key. After the action, the robot is holding the new key, and the old key is at the robot's current location.

- putdown <?curpos> <?key>: Allows the robot to put down a key it is holding if it is at a specific place. After the action, the robot's arm is empty, and the key is at that location.

A.2 Preliminary Experiment

The experimental results show that LLM's effectiveness in detecting PDDL semantic errors is limited, with an accuracy of 55.0%, a precision of 56.2%, a recall rate of 45.0%, an F1 score of 50.0%, and a ROC AUC of 55.0. ROC AUC indicates that the model is close to random performance, making it difficult to reliably distinguish between correct and incorrect PDDL domains. Below is the prompt used for LLMs to detect semantic errors in generated PDDL domains:

```
You are an expert in automated planning systems and PDDL semantics. Your task is to
    evaluate whether the LLM are physically accurate models of the world or whether
    they don't make sense by detecting semantic errors in generated PDDL domain.
You need carefully analyze the following PDDL domain by comparing it to the pddl
    domain description, evaluate whether the generated pddl domain contains SEMANTIC
     ERRORS in these key aspects:
1. Predicates consistency.
2. Action parameters validity.
3. Action preconditions completeness.
4. Action effects logical consistency.
5. Consistency with the description.

An example of semantic error would be:
1. Missing precondition constraints (e.g. executing "unlock-door" without holding a
    key).
2. Contradictory effects (e.g. both adding and deleting the same predicate).
3. Incorrect predicate arguments (e.g. reversed parameter order).

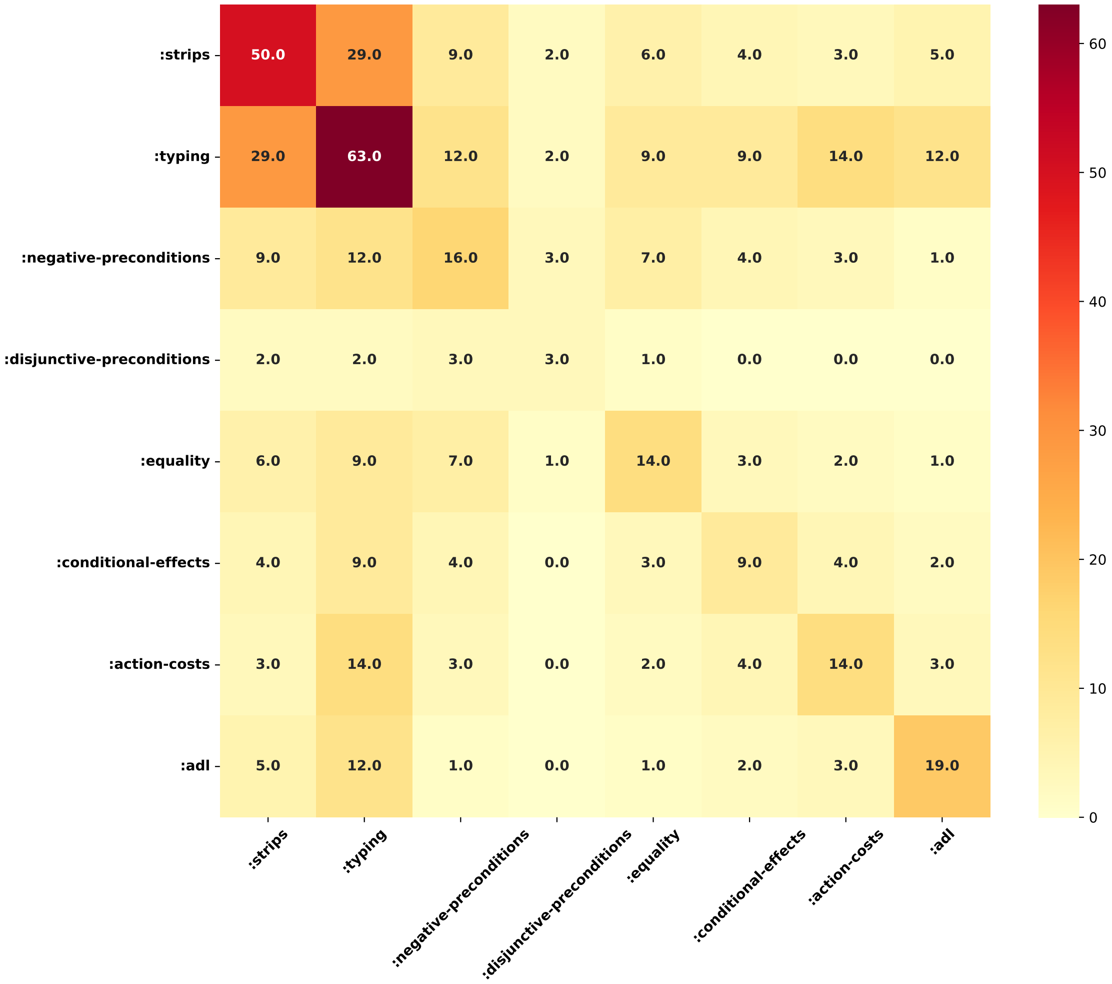
Output Format:
{
"evaluation": "yes/no",
"error_type": "[MissingPrecond|IncorrectEffect|MissingPredicate|...]",
"confidence": "high/medium/low",
"evidence": "<specific code segment>",
"justification": "<short justification>"
}

PDDL Description:
{PDDL_DESCRIPTION}

Generated PDDL:
{PDDL_DOMAIN}
```

A.3 More Details on Data Analysis

Figure 8 shows the co-occurrence of PDDL requirements across domains, highlighting that `:typing` and `:strips` are the most prevalent features.



	:strips	:typing	:negative-preconditions	:disjunctive-preconditions	:equality	:conditional-effects	:action-costs	:adl
:strips	50.0	29.0	9.0	2.0	6.0	4.0	3.0	5.0
:typing	29.0	63.0	12.0	2.0	9.0	9.0	14.0	12.0
:negative-preconditions	9.0	12.0	16.0	3.0	7.0	4.0	3.0	1.0
:disjunctive-preconditions	2.0	2.0	3.0	3.0	1.0	0.0	0.0	0.0
:equality	6.0	9.0	7.0	1.0	14.0	3.0	2.0	1.0
:conditional-effects	4.0	9.0	4.0	0.0	3.0	9.0	4.0	2.0
:action-costs	3.0	14.0	3.0	0.0	2.0	4.0	14.0	3.0
:adl	5.0	12.0	1.0	0.0	1.0	2.0	3.0	19.0

Figure 8: The co-occurrence matrix of requirements of TEXT2WORLD.

B More Details on Experiments

B.1 Evaluation Metrics

Levenshtein Ratio. The Levenshtein Ratio is a value between 0 and 1 that quantifies the similarity between two strings, such as a predicted PDDL domain and a golden PDDL domain. It is derived from the Levenshtein distance, which calculates the minimum number of character-level operations—insertions, deletions, or substitutions—needed to convert one string into the other. The ratio is then computed by dividing the Levenshtein distance by the length of the longer string, providing a measure of how closely the two strings match, where a value closer to 1 indicates high similarity and a value closer to 0 indicates significant differences.

Component-wise F1 Scores. The F1 score is mainly used to measure the similarity between the predicted PDDL domain and the golden PDDL domain, specifically including predicate F1 and action F1. The range of this score is from 0 to 1, which is the harmonic mean of precision and recall.

B.2 Prompt Examples

B.2.1 Error Correction

```
I would like you to serve as an expert in PDDL, assisting me in correcting erroneous
     PDDL code. I will provide you with the incorrect PDDL along with the error
    messages returned by the system. You should output your thought process firstly.
     You MUST enclose the COMPLETE corrected PDDL within ```pddl```.
Here are some hints to help you debug the pddl domain file:
1. You should start by checking if all the essential domain constructs or domain
    definition constructs are present. Commonly included domains comprise:
    a. :domain declaration to name the domain.
    b. :requirements to specify the PDDL features used in the domain.
    c. :types to define any object types for categorizing entities in the planning
        problem.
    d. :constants (if necessary) to declare any objects that remain static
        throughout the planning problems.
    e. :predicates to define the properties and relations between objects that can
        change over time.
    f. :functions (if necessary) to define numeric functions for more complex
        domains.
    g. :action definitions for each action that agents can perform, including
        parameters, preconditions, and effects.
2. You need to check the number of parameters of each actions.
3. Having :typing in the domain indicates that it uses a hierarchy to organize
    objects. Therefore, it's crucial to clearly list all object types related to the
     planning task in a :types section.
4. '-' should not appear in :types.



Round 0
Incorrect PDDL:
(:action clean-up
    :parameters (?robot - robot ?robotTile - tile ?tileToBeCleaned - tile)
    :precondition (and
        (robot-at ?robot ?robotTile)
        (up ?tileToBeCleaned ?robotTile)
        (clear ?tileToBeCleaned)
        (not (cleaned ?tileToBeCleaned))
    )
    :effect (and
        (cleaned ?tileToBeCleaned)
    )
)

(:action clean-down
    :parameters (?robot - robot ?robotTile - tile ?tileToBeCleaned - tile)
    :precondition (and
        (robot-at ?robot ?robotTile)
        (down ?tileToBeCleaned ?robotTile)
        (clear ?tileToBeCleaned)
        (not (cleaned ?tileToBeCleaned))
    )
    :effect (and
        (cleaned ?tileToBeCleaned)
    )
)

(:action up
    :parameters (?robot - robot ?robotTile - tile ?moveToNextTile - tile)
    :precondition (and
        (robot-at ?robot ?robotTile)
        (up ?moveToNextTile ?robotTile)
        (clear ?moveToNextTile)
    )
    :effect (and
        (not (robot-at ?robot ?robotTile))
        (robot-at ?robot ?moveToNextTile)
    )
)

(:action down
    :parameters (?robot - robot ?robotTile - tile ?moveToNextTile - tile)
```

```
    :precondition (and
        (robot-at ?robot ?robotTile)
        (down ?moveToNextTile ?robotTile)
        (clear ?moveToNextTile)
    )
    :effect (and
        (not (robot-at ?robot ?robotTile))
        (robot-at ?robot ?moveToNextTile)
    )
)

(:action right
    :parameters (?robot - robot ?robotTile - tile ?moveToNextTile - tile)
    :precondition (and
        (robot-at ?robot ?robotTile)
        (right ?moveToNextTile ?robotTile)
        (clear ?moveToNextTile)
    )
    :effect (and
        (not (robot-at ?robot ?robotTile))
        (robot-at ?robot ?moveToNextTile)
    )
)

(:action left
    :parameters (?robot - robot ?robotTile - tile ?moveToNextTile - tile)
    :precondition (and
        (robot-at ?robot ?robotTile)
        (left ?moveToNextTile ?robotTile)
        (clear ?moveToNextTile)
    )
    :effect (and
        (not (robot-at ?robot ?robotTile))
        (robot-at ?robot ?moveToNextTile)
    )
)
Error Information:
ParsingError: line 1:1 mismatched input ':action' expecting 'define'
Corrected PDDL:
```

B.2.2 Zero-Shot Prompt

```
You are tasked with converting a given Planning Domain Definition Language (PDDL)
    domain description into its corresponding formal PDDL domain. The description
    will outline the essential components of the domains.
Your output should be a well-structured PDDL domain that accurately represents the
    given description, adhering to the syntax and semantics of PDDL.
Your output pddl domain must be enclosed in ```pddl```.

You need to generate the corresponding domain pddl for the following description.

PDDL Domain Description:
### General
This domain is designed for a robot tasked with cleaning floor tiles. The robot can
    move in four directions (up, down, right, left) relative to its current position
     on a grid of tiles. The goal is to clean all the specified tiles by moving to
    them and performing a cleaning action.

### Types
- **robot**: Represents the robot that performs the cleaning.
- **tile**: Represents the individual tiles on the floor that may need to be cleaned
    .

### Predicates
- **(robot-at ?robot - robot ?robotTile - tile)**: Indicates that the robot is
    currently at a specific tile.
- **(up ?tileAbove - tile ?tileBelow - tile)**: Indicates that one tile is directly
    above another.
- **(down ?tileBelow - tile ?tileAbove - tile)**: Indicates that one tile is
    directly below another.
```

````
- **(right ?tileOnRight - tile ?tileOnLeft - tile)**: Indicates that one tile is
    directly to the right of another.
- **(left ?tileOnLeft - tile ?tileOnRight - tile)**: Indicates that one tile is
    directly to the left of another.
- **(clear ?clearedTile - tile)**: Indicates that a tile is clear and robot can move
     there.
- **(cleaned ?cleanedTile - tile)**: Indicates that a tile has been cleaned.

### Actions
- **clean-up <?robot> <?robotTile> <?tileToBeCleaned>**: Allows the robot (?robot)
    to clean a tile (?tileToBeCleaned) that is directly above its current position
    (?robotTile).

- **clean-down <?robot> <?robotTile> <?tileToBeCleaned>**: Allows the robot (?robot)
     to clean a tile (?tileToBeCleaned) that is directly below its current position
    (?robotTile).

- **up <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a tile
     (?moveToNextTile) directly above its current position (?robotTile).

- **down <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a
    tile (?moveToNextTile) directly below its current position (?robotTile).

- **right <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a
    tile (?moveToNextTile) directly to the right of its current position (?robotTile
    ).

- **left <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a
    tile (?moveToNextTile) directly to the left of its current position (?robotTile)
    .
PDDL Domain:
Let's think step by step.
````

B.2.3 Few-Shot Prompt

~~~
You are tasked with converting a given Planning Domain Definition Language (PDDL)
    domain description into its corresponding formal PDDL domain. The description
    will outline the essential components of the domains. Your output should be a
    well-structured PDDL domain that accurately represents the given description,
    adhering to the syntax and semantics of PDDL.
Your output must strictly adhere to the format exemplified below.
Here are some examples:

Example 0:
## PDDL Domain Description
### General
You are a robot equipped with a gripper mechanism, designed to move and manipulate
    balls between different rooms. The domain focuses on the robot's ability to
    navigate rooms, pick up balls, and drop them in designated locations.
### Types
- **room**: Represents the different rooms within the environment.
- **ball**: Represents the objects that the robot can pick up and move.
- **gripper**: Represents the robot's mechanism for holding balls.
### Predicates
- **(at-robby ?r - room)**: Indicates that Robby, the robot, is currently in room ?r
    .
- **(at ?b - ball ?r - room)**: Indicates that ball ?b is located in room ?r.
- **(free ?g - gripper)**: Indicates that the gripper ?g is not currently holding
    any ball.
- **(carry ?o - ball ?g - gripper)**: Indicates that the gripper ?g is carrying ball
     ?o.
### Actions
- **move <?from> <?to>**: Allows Robby to move from one room to another.
- **pick <?obj> <?room> <?gripper>**: Enables Robby to pick up a ball in a room
    using its gripper.
- **drop <?obj> <?room> <?gripper>**: Allows Robby to drop a ball it is carrying
    into a room.

## PDDL Domain
```pddl
~~~

~~~
(define (domain gripper-strips)
        (:types
                room - object
                ball - object
                gripper - object
                )
   (:predicates
                (at-robby ?r - room)
                (at ?b - ball ?r - room)
                (free ?g - gripper)
                (carry ?o - ball ?g - gripper))
   (:action move
       :parameters  (?from - room ?to - room)
       :precondition (and (at-robby ?from))
       :effect (and  (at-robby ?to)
                     (not (at-robby ?from))))
   (:action pick
       :parameters (?obj - ball ?room - room ?gripper - gripper)
       :precondition  (and
                            (at ?obj ?room) (at-robby ?room) (free ?gripper))
       :effect (and (carry ?obj ?gripper)
                    (not (at ?obj ?room))
                    (not (free ?gripper))))
   (:action drop
       :parameters  (?obj - ball ?room - room ?gripper - gripper)
       :precondition  (and
                            (carry ?obj ?gripper) (at-robby ?room))
       :effect (and (at ?obj ?room)
                    (free ?gripper)
                    (not (carry ?obj ?gripper)))))
```


Example 1:
## PDDL Domain Description
### General
This domain represents a simplified version of the classic "blocks world" problem,
    where a robot arm can stack and unstack blocks. The domain includes actions for
    picking up blocks from the table, putting down blocks onto the table, stacking
    blocks on top of each other, and unstacking them. The goal is to manipulate the
    blocks to achieve a specified configuration.
### Predicates
(clear ?x): Indicates that there is no block on top of block ?x, making it
    accessible for stacking or picking up.
(on-table ?x): Indicates that block ?x is directly on the table.
(arm-empty): Indicates that the robot's arm is not holding any block.
(holding ?x): Indicates that the robot's arm is currently holding block ?x.
(on ?x ?y): Indicates that block ?x is directly on top of block ?y.
### Actions
- **pickup <?ob>**: Picks up an object (?ob) from the table.
- **putdown <?ob>**: Puts down an object (?ob) onto the table.
- **stack <?ob> <?underob>**: Stacks an object (?ob) on top of another object (?
    underob), making the robot arm empty.
- **unstack <?ob> <?underob>**: Unstacks an object (?ob) from another object (?
    underob), making the robot arm no longer empty.

## PDDL Domain:
```pddl
(define (domain blocksworld)
  (:requirements :strips)
(:predicates (clear ?x)
             (on-table ?x)
             (arm-empty)
             (holding ?x)
             (on ?x ?y))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
~~~

~~~~
               (not (arm-empty))))

(:action putdown
  :parameters  (?ob)
  :precondition (holding ?ob)
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))

(:action stack
  :parameters  (?ob ?underob)
  :precondition (and (clear ?underob) (holding ?ob))
  :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob)
               (not (clear ?underob)) (not (holding ?ob))))

(:action unstack
  :parameters  (?ob ?underob)
  :precondition (and (on ?ob ?underob) (clear ?ob) (arm-empty))
  :effect (and (holding ?ob) (clear ?underob)
               (not (on ?ob ?underob)) (not (clear ?ob)) (not (arm-empty)))))
```

You need to generate the corresponding domain pddl for the following description.

## PDDL Domain Description
### General
This domain is designed for a robot tasked with cleaning floor tiles. The robot can move in four directions (up, down, right, left) relative to its current position on a grid of tiles. The goal is to clean all the specified tiles by moving to them and performing a cleaning action.

### Types
- **robot**: Represents the robot that performs the cleaning.
- **tile**: Represents the individual tiles on the floor that may need to be cleaned.

### Predicates
- **(robot-at ?robot - robot ?robotTile - tile)**: Indicates that the robot is currently at a specific tile.
- **(up ?tileAbove - tile ?tileBelow - tile)**: Indicates that one tile is directly above another.
- **(down ?tileBelow - tile ?tileAbove - tile)**: Indicates that one tile is directly below another.
- **(right ?tileOnRight - tile ?tileOnLeft - tile)**: Indicates that one tile is directly to the right of another.
- **(left ?tileOnLeft - tile ?tileOnRight - tile)**: Indicates that one tile is directly to the left of another.
- **(clear ?clearedTile - tile)**: Indicates that a tile is clear and robot can move there.
- **(cleaned ?cleanedTile - tile)**: Indicates that a tile has been cleaned.

### Actions
- **clean-up <?robot> <?robotTile> <?tileToBeCleaned>**: Allows the robot (?robot) to clean a tile (?tileToBeCleaned) that is directly above its current position (?robotTile).

- **clean-down <?robot> <?robotTile> <?tileToBeCleaned>**: Allows the robot (?robot) to clean a tile (?tileToBeCleaned) that is directly below its current position (?robotTile).

- **up <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a tile (?moveToNextTile) directly above its current position (?robotTile).

- **down <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a tile (?moveToNextTile) directly below its current position (?robotTile).

- **right <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a tile (?moveToNextTile) directly to the right of its current position (?robotTile).

- **left <?robot> <?robotTile> <?moveToNextTile>**: Moves the robot (?robot) to a
~~~~

```
    tile (?moveToNextTile) directly to the left of its current position (?robotTile)
    .
## PDDL Domain
```

C More Details on Analysis

C.1 Overall

Table 3: Distribution of error types for `claude-3.5-sonnect` and `deepseek-r1` on TEXT2WORLD under few-shot setting.

	Claude-3.5-sonnect		**Deepseek-r1**	
Error Type	**Proportion (%)**	**Number**	**Proportion (%)**	**Number**
Correct	23.76	24	15.84	16
Syntax Error	11.88	12	9.90	10
Semantic Error	64.36	65	74.26	75
All	100.00	101	100.00	101

The overall distribution for syntax errors and semantic errors is presented in Table 3.

C.2 Syntax Error

Table 4: Distribution of Syntax Errors for `claude-3.5-sonnect` in PDDL Generation (Total Samples: 66, a task may have 1 to 4 samples.)

Syntax Error	**Explanation**	**Proportion (%)**
UndefinedDomainName	Missing mandatory `(define (domain ...))` declaration in PDDL header	33.33
IncorrectParentheses	Invalid empty/mismatched parentheses	3.03
UndefinedConstant	Reference to undeclared constants in predicates or actions	13.64
MissingRequirements	Absence of required PDDL extension declarations (e.g., `:action-costs`)	22.73
UndefinedType	Undeclared parent type in hierarchical type definitions	18.18
UnsupportedFeature	Use of parser-incompatible language features (e.g., `either` types)	3.03
TypeMismatch	Parameter type conflict with declared type constraints	1.52
UndefinedVariable	Undeclared variables in action preconditions/effects	1.52
DuplicateDefinition	Multiple declarations of identical domain elements	3.03

Table 5: Distribution of Syntax Errors for `deepseek-r1` in PDDL Generation (Total Samples: 77, a task may have 1 to 4 samples.)

Syntax Error	**Explanation**	**Proportion (%)**
UndefinedDomainName	Missing mandatory `(define (domain ...))` declaration in PDDL header	0.00
IncorrectParentheses	Invalid empty/mismatched parentheses	33.77
UndefinedConstant	Reference to undeclared constants in predicates or actions	6.49
MissingRequirements	Absence of required PDDL extension declarations (e.g., `:action-costs`)	23.38
UndefinedType	Undeclared parent type in hierarchical type definitions	20.78
UnsupportedFeature	Use of parser-incompatible language features (e.g., `either` types)	3.90
TypeMismatch	Parameter type conflict with declared type constraints	1.30
UndefinedVariable	Undeclared variables in action preconditions/effects	5.19
DuplicateDefinition	Multiple declarations of identical domain elements	5.19

The distribution and detailed explanation of each syntax error type are presented in Table 4 and Table 5.

C.3 Semantic Error

The distribution and detailed explanation of each semantic error type are presented in Table 6 and Table 7.

Table 6: Distribution of Semantic Errors for `claude-3.5-sonnect` in PDDL Generation (Total Samples: 91, a task may have multiple semantic errors.)

Semantic Error	Explanation	Proportion (%)
DisobeyDescription	Direct violation of semantic requirements explicitly stated in the task description.	**14.29**
IncorrectPredicate	Incorrect or missing the declaration of predicates.	6.59
IncorrectAction	Incorrect or missing the declaration of actions.	7.69
IncompleteModeling	Incomplete world modeling compared to basic requirements.	**58.24**
IncorrectPrecondition	The precondition of the action is deficient or incorrect.	29.67
IncorrectEffect	The effect of the action is deficient or incorrect.	28.57
RedundantSpecifications	Predicted domain includes superfluous preconditions or effects.	**17.58**
RedundantPrecondition	Predicted domain includes superfluous preconditions.	10.99
RedundantEffect	Predicted domain includes superfluous effects.	6.59
SurfaceDivergence	Surface variations preserving semantic equivalence with ground truth.	**9.89**

Table 7: Distribution of Semantic Errors for `deepseek-r1` in PDDL Generation (Total Samples: 105, a task may have multiple semantic errors.)

Semantic Error	Explanation	Proportion (%)
DisobeyDescription	Direct violation of semantic requirements explicitly stated in the task description.	**12.38**
IncorrectPredicate	Incorrect or missing the declaration of predicates.	6.67
IncorrectAction	Incorrect or missing the declaration of actions.	5.71
IncompleteModeling	Incomplete world modeling compared to basic requirements.	**66.67**
IncorrectPrecondition	The precondition of the action is deficient or incorrect.	34.29
IncorrectEffect	The effect of the action is deficient or incorrect.	32.38
RedundantSpecifications	Predicted domain includes superfluous preconditions or effects.	**14.29**
RedundantPrecondition	Predicted domain includes superfluous preconditions.	10.48
RedundantEffect	Predicted domain includes superfluous effects.	3.81
SurfaceDivergence	Surface variations preserving semantic equivalence with ground truth.	**6.67**

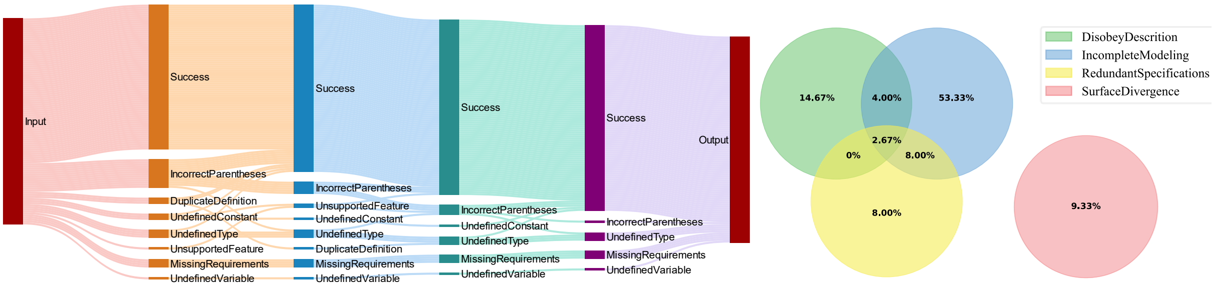


Figure 9: *Left*: The distribution of syntax error types for `deepseek-r1` during the progression of correction. *Right*: The distribution of semantic error types.

D More Experimental Results

D.1 Experimental Results with Concrete Description

Table 8: Performance comparison of different LLMs on TEXT2WORLD using concrete domain description. EC_k denotes the setting where models are allowed k correction attempts (EC_0: zero-shot without correction, EC_3: with 3 correction attempts).

Model Family	Version	EXEC. ↑		SIM. ↑		$\mathbf{F1}_{\text{PRED}}$ ↑		$\mathbf{F1}_{\text{PARAM}}$ ↑		$\mathbf{F1}_{\text{PRECOND}}$ ↑		$\mathbf{F1}_{\text{EFF}}$ ↑	
		$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$	$\mathbf{EC_0}$	$\mathbf{EC_3}$
GPT-4	`gpt-4o`	60.4	75.2	90.7	90.3	59.4	71.8	57.1	69.1	55.3	65.1	54.1	65.2
GPT-3.5	`turbo-0125`	53.5	68.3	89.0	88.7	52.9	66.7	50.3	64.6	45.1	58.0	46.5	59.9
CLAUDE-3.5	`sonnet`	64.4	84.2	84.7	77.6	64.4	80.7	55.0	67.5	53.3	65.0	53.3	64.8
LLAMA-2	`7b-instruct`	0.0	0.0	48.4	32.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	`70b-instruct`	0.0	0.0	53.5	52.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LLAMA-3.1	`8b-instruct`	0.0	1.0	84.1	83.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	`70b-instruct`	1.0	1.0	89.7	85.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DEEPSEEK	`deepseek-v3`	58.4	80.2	90.1	89.3	58.1	76.4	56.2	73.5	53.4	66.0	53.5	67.6