Plant in Cupboard, Orange on Table, Book on Shelf
Benchmarking Practical Reasoning and Situation Modelling in a Text-Simulated Situated Environment

Jonathan Jordan[1], Sherzod Hakimov[1], David Schlangen[1,2]

[1]Computational Linguistics, Department of Linguistics
University of Potsdam, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
`firstname.lastname@uni-potsdam.de`

Abstract

Large language models (LLMs) have risen to prominence as "chatbots" for users to interact via natural language. However, their abilities to capture common-sense knowledge make them seem promising as language-based planners of situated or embodied action as well. We have implemented a simple text-based environment — similar to others that have before been used for reinforcement-learning of agents — that simulates, very abstractly, a household setting. We use this environment and the detailed error-tracking capabilities we implemented for targeted benchmarking of LLMs on the problem of practical reasoning: Going from goals and observations to actions. Our findings show that environmental complexity and game restrictions hamper performance, and concise action planning is demanding for current LLMs.

1 Introduction

Theoretical reasoning, which involves deriving factual conclusions from given premises, has been extensively studied in the context of large language models (LLMs) (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022). However, there has been less work focusing on practical reasoning capabilities (Wallace and Kiesewetter, 2024), where models must generate actions based on environmental observations. While theoretical reasoning can be evaluated using reference propositions, practical reasoning presents unique challenges because the evaluation requires observing the consequences of actions in an environment. There is significant interest in using LLMs for embodied AI tasks in robotics and simulations. (Ichter et al., 2022). Many of the existing benchmarks (e.g., WebArena (Zhou et al., 2024), ALFRED (Shridhar et al., 2021), AI2-THOR (Kolve et al., 2022), Balrog Arena (Paglieri et al., 2024)) include complex environments or require computational resources to execute vision-related tasks.
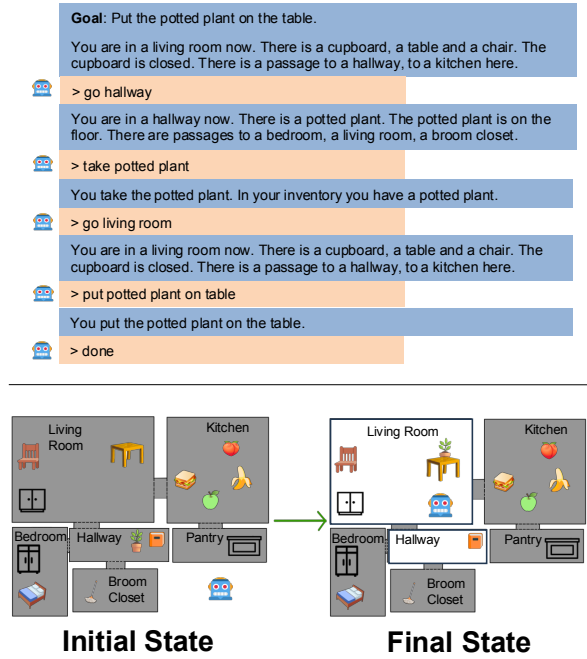

Figure 1: Sample representation of an episode in our game. The agent 🤖 (an LLM) is given a task and randomly assigned to some room (living room). The environment provides feedback for every action (go, take, put, etc.) of an agent. The top part is how the game is played in a textual world. The bottom part is the visual representation of initial and final state, also indicating the situation knowledge gained by the agent.

In contrast, text-based environments such as TextWorld (Côté et al., 2019), where a situation is described textually and updated in response to textual commands, make it possible to study practical reasoning abilities that go from goal and situation description to goal-directed action in greater isolation. We built an even more simplified environment (called *AdventureGame*) for testing situated language environment understanding, spatial navigation, practical and common-sense reasoning, and instruction following abilities.

Figure 1 (top) shows an example of an interaction, while the bottom part is the visual represen-

tation of what has changed going from an initial state to the final one.

This interaction can be thought of as between a robot's "brain" and its perceptual and actuating parts. Our contributions are as follows:

- A light-weight and extensible IF environment that includes in-depth fine-grained tracking of the world state, player observations and interaction failures.
- A set of experimental instances condensing an established object delivery task in a typical home setting, with varied levels of complexity and task demands of a situated interaction.
- Quantitative comparison of recent large language models (LLM) of varying sizes.
- In-depth analysis of contributing factors for high-performing models.
- Qualitative analysis of model behaviour, examining good interaction and common failure cases.

2 Related Work

Interactive Fiction environments have been used to *train and compare agents*, starting in the field of Reinforcement Learning, using classic IF formats and interpreters (Côté et al. (2019), Hausknecht et al. (2020)). These environments combine a specific set of agent challenges: Partial observability, large world state and action space, exploration and common sense reasoning, in addition to the text-only interaction. It has been shown that performance in text-only IF environments is transferable to embodied environments (Shridhar et al. (2021), Jansen (2021)). However, the frameworks established with these works have convoluted code dependencies that hinder reproduction.

With the rise of transformer LLMs (Vaswani et al., 2017), IF environments have also been used to *compare LLM performance* in regards to their world modelling, task solving and planning capabilities (Wang et al. (2022), Tan et al. (2023), Tsai et al. (2023), Ma et al. (2024), Gioacchini et al. (2024)). These works find that direct interaction with IF environments is challenging for LLMs, although they have a substantial advantage regarding common sense world knowledge compared to RL agents. Few works, however, have provided fine-grained success and failure metrics, hindering interpretation.

Recently, IF-based environments have been used to compare agent systems that incorporate LLMs to leverage their common sense world knowledge and reasoning capabilities (Wang et al. (2022), Basavatia et al. (2024), i.a.). Theses agent systems involve extensive augmentation around LLMs, with Retrieval Augmented Generation, multi-stage prompting and external planners and verifiers to guide the LLM (Park et al. (2023), Valmeekam et al. (2023), Tikhonov (2024), Jansen et al. (2024), Li et al. (2025), i.a.). A number of works have LLM agents generate entire solution plans based on fully observable world states without direct interaction, which makes automated evaluation easier but foregoes examining incremental world modelling and exploration (Xie et al. (2023), Silver et al. (2022), i.a.).

AdventureGame is filling a gap in assessing the performance of unassisted LLMs, tapping into claimed capabilities with minimal guidance. *AdventureGame* provides observations of action outcomes to the LLM directly each turn, combining planning and execution.

3 *AdventureGame*: IF Environment

In this section, we describe the details of the game. A simplified illustration is given in Figure 2, where an LLM controls an agent to perform the given task. We implemented the game using the clembench framework (Chalamalasetti et al., 2023) (prompts are provided in Appendix 10).

3.1 Task Definition

Interactive Fiction games can be considered partially observable Markov Decision processes (Kaelbling et al., 1998), with the player having only partial information about the game's world state from textual observations. The player must first discover all task-relevant locations and objects, making exploration as much a part of solution strategies as effectively executing a plan for manipulating objects to reach the game's goals. To do so successfully, the player has to model environment states (like room connections and locations of objects), as well as the large action space and state transitions resulting from actions, especially over multiple turns.

The task is defined by the tuple $\langle G, S, A, O, T \rangle$ with a set of goal states G, state space S, valid action space A, observation space O (including IF interpreter feedback), and transition function $T : S \times A \to S$.

An agent with policy π makes predictions at time step t based on goals in G and memory $m_t =$
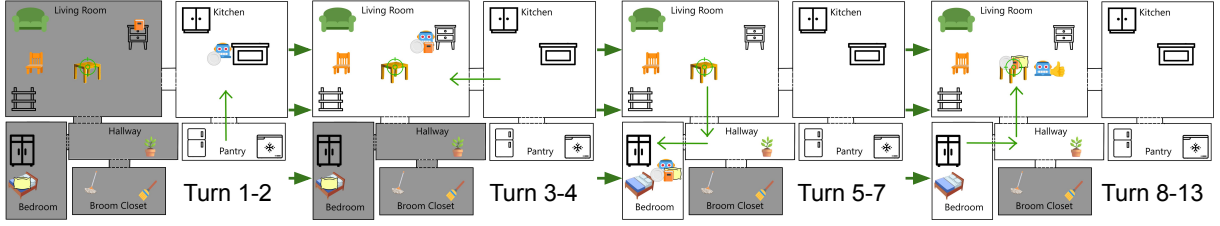
Figure 2: Simplified illustration of *AdventureGame* interaction. The agent 🤖 is controlled by an LLM. The task is *put the plate, book and pillow on the table* (marked by green crosshair). The agent starts in the pantry. Unexplored rooms are gray. The agent first goes to the kitchen, *takes the plate from the counter* (Turn 1-2), then *goes to the living room, takes the book* (Turn 3-4). Next, it *goes to the hallway*, and from there *to the bedroom where it picks up the pillow* (Turn 5-7), then it *returns to the living room, puts the carried objects on the table* (Turn 8-13).

$\{o_j, a_j, o_{j+1}, a_{j+1}, ...o_t\}, 0 \leq j < t$, which is a sequence of actions and observations. This agent trajectory $\tau = [s_0, a_0, s_1, a_1, ...s_t]$ is formulated by policy and environmental state transitions as below where a time step t is one turn of the game:

$$p_\pi(\tau) = p(s_0) \prod_{t=0}^{T} \pi(a_t|G, s_t, m_t)\tau(s_{t+1}|s_t, a_t)$$

All state transitions are deterministic, so only the actions generated by the LLM as text commands determine the agent's trajectory. Observations are likewise deterministic. We assume that the language model models an underlying policy, tapping into the capabilities under examination.

3.2 Interaction

World state, representing a temporary subset of S, is stored as a set of fact tuples, describing both mutable states of entities and immutable states. Mutable states are at, in, on, open and closed. Immutable states describe entity types and categories and are used as conditions for actions (see Table 3).

Actions are defined in the PDDL (Fox and Long, 2003) format, covering the state changes they enact (representing T). The action definition also contains a Lark parser grammar snippet that is used to form the parse-able input command grammar, feedback templates for success and individual failures (covering O) and a Clingo (Gebser et al., 2017) encoding snippet of the state changes of the action for optimal solution solving (representing T as well).

The player is not given a list of currently available actions but rather has to model the action space itself. Movement is only allowed to connected rooms (unlike in Basavatia et al. (2024) and others, which allow "teleportation").

Turn Limit: The interaction is limited to fifty turns, and reaching the limit is recorded as aborting that episode.

Formatting: An output produced by LLM has to follow this format (current command followed by the next commands):

```
> "COMMAND"
Next actions: "COMMANDS"
```

The basic task type requires only the first command line, while the planning task requires both outputs (see Section 4.1).

Parsing & State Change: The interpreter attempts to parse the command, and if it passes, the corresponding action is performed in the resolution phase. State change conditions are checked, and any resolution failure stops the process. If state change conditions are fulfilled, the game world state is updated accordingly, removing and adding facts in the world state. All changes in the world state are recorded. Lastly, the interpreter checks if changed states are part of the goal state set G and returns achieved goal states G_a, textual feedback $o \in O$ and any failure that might have occurred in processing the input command to be recorded.

For the planning variant, this procedure is performed subsequently for each input command in the "Next actions:" list (corresponding to τ_P) until a failure is encountered. World state history is used to revert any state changes resulting from a planned action sequence.

An episode is ended by the player using the 'done' action or reaching the turn limit. Once the episode ends, goal states achieved as of the last turn are recorded - meaning that goal states achieved intermittently are not considered for metrics.

Exploration Tracking: All commands, state changes, observed locations and objects are recorded for each turn, including errors while executing the commands. We label each action (in-

spired by Kirsh and Maglio (1994)) for being *epistemic* and *pragmatic*. An action is *epistemic* when it improves a player's perception of the game situation without directly progressing towards the goal[1] and *pragmatic* when the action directly works towards reaching the goal. Then, we calculate *epistemic gain* by counting how many world state facts the player newly observes as a result.

3.3 Challenges & Examined Capabilities

The task includes various challenges and their accompanying capabilities that are targeted here, purely in text. Despite the environment being abstract compared to physically embodied agents, these capabilities are required for autonomous performance regardless of the modality.

World state modelling: The agent must correctly recall objects' states and locations over multiple turns under partial observability by consecutively integrating the observations provided by the game.

Situated action selection: Another essential capability is selecting appropriate actions in a specific situation from many options, especially without any guidance to limit this action space.

Common sense reasoning & world knowledge: The agent should perform action selection, and locating objects in their ordinary locations requires world knowledge and reasoning.

Spatial reasoning & exploration: Navigation relies on spatial modelling of the game environment, especially for going through multiple locations. Sufficient exploration of the environment is necessary to correctly fulfil the task, which entails a certain level of trial and error, with actions that are not immediately useful to further the task goal.

Self-correction: The capability to revise and correct an assumed world model in the face of new observations is crucial to any situated task solving. An agent should be capable of verifying the correctness of its world model through its interactions with the world and actively do so.

4 Experimental Setup

4.1 Game Instances

Different instances of the game are generated using the Clingo Answer Set Programming library[2],

which wraps the Clingo solver (Gebser et al., 2017). We create challenging instances by varying three factors for the experiments, as given below.

Task variants: we have two variants of the task: basic & planning. The core task in our experiments is a *home delivery task* in which the agent is expected to deliver three objects to target receptacles in a common home environment. For the *planning* variant, models are prompted also to list the next actions. Our hypothesis is that the *planning* variant is more challenging because it requires knowing more about the environment, and being able to adapt across turns. We want to analyse whether models have any strategy or if they simply invent arbitrary plans.

Delivery difficulty: it has two levels: easy & hard. The *easy level* means that goal objects are not inside closed containers (easily accessible, immediately observable) and are located near the target receptacle in the initial world state. The *hard level* means that goal objects are initially hidden in closed containers, each needs to be delivered to a different target, and there are longer paths between initial goal object locations and their targets. Our hypothesis is that it is challenging for models to navigate multi-step actions to reach a goal because hidden objects require epistemic actions, far away objects require keeping the objective in mind over several steps (increases the number of steps to reach goals).

Inventory limit: by default, the number of objects the agent can carry is unlimited. We create another level by setting the limit to *two*. We want to analyse whether models lean towards strategic use of the resource "inventory" and being efficient, when they have limit.

We have eight experiments (permutation of task types, object difficulties, inventory limit) (16 instances in each), corresponding to 128 instances.

4.2 Metrics

Framework-specific metrics: The clembench framework includes two main metrics: *Played & Quality Score*. The game finishes successfully only when a model produces "> done" as the last action and all goals have been achieved. The game is *aborted* when a model does not follow formatting requirements (see Section 3.2) or reaches the *maximum turn limit*, which is 50. Played is the ratio of instances that were not aborted. The quality score measures how many episodes have all their goal states reached at the end. Producing the "> done"

---

[1]Kirsh and Maglio's example is moving a Tetris tile to the leftmost position to be sure about its position instead of moving it towards the location that is most beneficial to put it down in.

[2]https://potassco.org/clingo/python-api/current/clingo/

action command without achieving all goal states is considered a *lost* episode. In cases where all goal states are achieved and "> done" is also generated, then the episode is *successful*.

Finally, to rank the benchmarked models, the framework includes a metric called *clemscore*, the macro-average quality score multiplied by the macro-average proportion of played games across all experiments.

Game-specific metrics: We have a specific metric to keep track of achieved goals. It is the ratio between achieved goal states G_a and all goal states G: Goal Success Rate (GSR) = $\frac{|G_a|}{|G|} \times 100$.

For the planning variant of the task, we have the plan *viability* metric. It is calculated as the fraction of successfully processed plan actions a_t^+ over the number of total plan actions a_t in turn t: $v_t = \frac{|a_t^+|}{|a_t|}$. Then, we calculate the average plan viability in an episode as the average of each turn (except the first turn): $viability = \frac{\sum_{t=1}^{T} v_t}{T}$.

4.3 Evaluated Models

We evaluated open-weight and commercial models (with *temp=0*). We included recent commercial models such as: *o3-mini* (Jan '25), *GPT-4o* (Aug '24) *Claude-3-5* (Sonnet, Oct '24), and *Gemini-2.0-Flash* (Feb '25). We also included recent open-weight models: *Llama-3.1* (8B, 70B, 405B) (Grattafiori et al., 2024), *Llama-3.3* (70B), *Qwen2* (72B) (Yang et al., 2024), *Qwen2.5* (Coder-32B, 72B, Max) (Qwen et al., 2025), *Sky-T1-32B* (NovaSky-Team, 2025) and *Deepseek-v3* (DeepSeek-AI et al., 2024). We used the APIs of the respective commercial models. We ran open-weight models on two NVIDIA A100 GPUs. Deepseek-v3, Llama-3.1-405B, and Qwen-Max were run via the OpenRouter API.

5 Results

5.1 Overall Comparison

Table 1 shows the main scores for the benchmarked models. Larger models achieve higher quality scores and better conform to the prompted output format. Most commercial models achieve higher scores than open-weight models (11 points between *Claude-3.5* and *Llama-3.1-70B*) except *Gemini-2.0*, which scores lower on both *Played* and *Quality Score*. Another observation is that all high-ranking models can play the game (follow instructions) but lack performance in solving the task. Next, we

Model	clem score	% Played	Quality Score	Goal Rate
o3-mini	63.1	85.9	73.4	79.7
Claude-3.5	62.5	93.0	67.2	76.3
GPT-4o	50.2	94.5	53.1	71.6
Qwen-max	47.4	90.6	52.3	68.8
Llama-3.1-70B	41.4	94.5	43.8	62.8
DeepSeek-V3	40.3	87.5	46.1	64.8
Llama-3.3-70B	39.5	95.3	41.4	60.2
Llama-3.1-405B	31.7	71.1	44.5	62.2
Gemini-2.0	25.0	68.0	36.7	50.5
Qwen2.5-32B	22.4	60.9	36.7	57.6
Qwen2-72B	18.8	58.6	32.0	49.5
Qwen2.5-72B	16.0	58.6	27.3	49.2
Llama-3.1-8B	13.0	46.1	28.1	40.1
Sky-T1-32B	5.7	28.9	19.5	31.8

Table 1: Overall benchmark scores for models.

analyse the cases deeper to uncover which factors contribute to low scores.

5.2 In-depth Analysis

In this section, we analyse different factors that contributed to certain models' better performance than others and investigate others.

5.2.1 Difficulty Levels

We compared five high-ranking models across all experiments (see Table 2). For smaller models, both higher complexity and limited inventory lead to worse performance, with the effects of both compounding. Specifically for the *home delivery* task, *Claude-3.5* and *o3-mini* perform equally for the *easy* and *hard* levels. At the same time, other models tend to struggle more with the *hard* episodes. One assumption on better performance on *hard* level (goal objects are hidden inside containers) is that the more complex tasks inherently require more exploration, and multiple target receptacles are less semantically similar to others. Adding an inventory limit (max two objects) resulted in mixed outcomes. Some models got better results (*o3-mini* on *hard*, *GPT-4o* on *easy*), but the general trend is that adding a limit on the inventory was more challenging.

5.2.2 Action Planning

The *planning* variants of experiments are found to be more challenging than the *home delivery* one, especially for smaller models (see Table 2). The models not only have to generate the immediate subsequent action but also the following actions. For instance, models generate generic plans such as "find apple and table" (where the goal statement is "Put the apple on the table") in the early turns be-
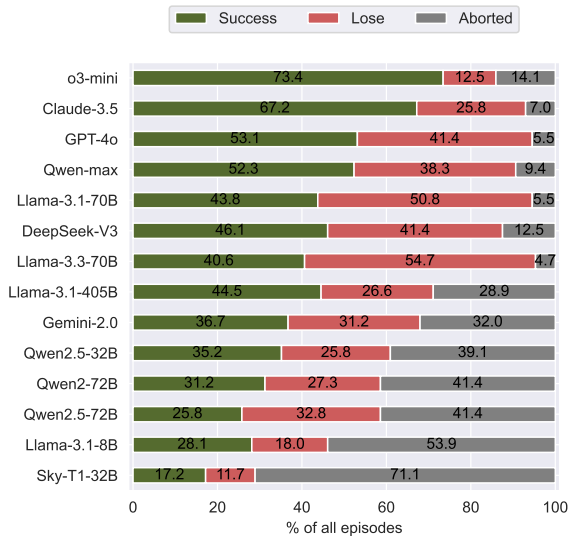
Figure 3: Successful, lost and aborted episode ratios.

cause, at those stages, the environment has not been fully explored. Such plans are not parsable and result in lower plan viability. In the later turns, the plan viability scores go higher a bit (e.g., *Claude-3.5*) but still fluctuate within a certain margin (see Figure 12). Overall, models struggled with this task because they moved to a more abstract level of planning ("find kitchen") due to the lack of world information in the initial phases.
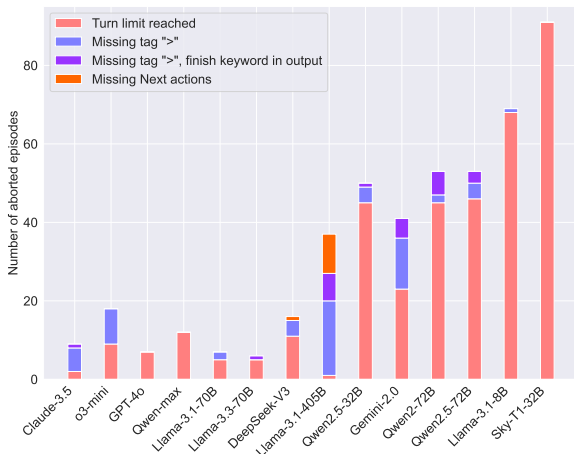

Figure 4: Distribution of causes for aborted episodes.

5.2.3 Successful, Lose & Aborted Episodes

Figure 3 shows the distribution of episodes across Success (all goal states are reached), Lose (if any goal state is not reached), or Aborted (formatting was not followed through or turn limit reached). The best performing two models *Claude-3.5* and *o3-mini*, solve most of the episodes (>=60%) and have lower numbers of lost episodes. The main
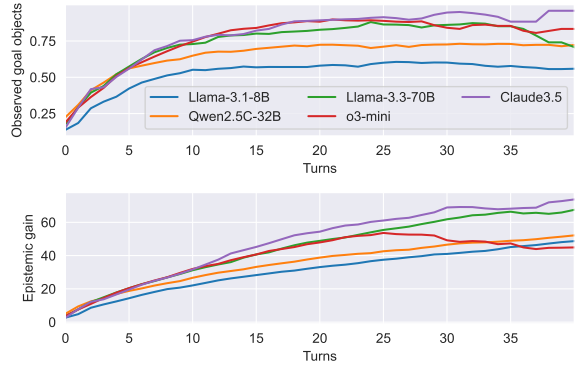

Figure 5: Average observed goal objects ratio and average cumulative effective epistemic gain over all episodes.

contributing factor for other models is that they either lose episodes or have them aborted due to violating instruction formatting requirements. Figure 4 shows the distribution of causes for aborted episodes. The most common issue for *Claude-3.5* and *o3-mini* is that they missed the tag ">" before an action statement. Some of the other models struggle with finishing the game by producing "> done" and reaching the maximum turn limit (50), or *Llama3.1-405B* did not generate the "Next actions" for the planning experiment. Overall, we can conclude that the main factor for the aborted cases is *models not knowing when to stop the game*, i.e., failing to recognise that the current situation is a goal state.

5.2.4 Exploration

All tested models have issues with sufficiently exploring the game environment: Rooms containing task-relevant objects are not entered, and containers containing task-relevant objects are not opened. Figure 5 (top) shows the average ratio of *observed goal entities* per turn. We can see that high-performing models such as *o3-mini* and *Claude-3.5* explore more and discover more required objects compared to smaller models such as *Llama-3.1-8B* or *Qwen2.5C-32B*. For smaller models, we assume this is due to a lack of spatial modelling and the inability to handle long input contexts. In comparison, larger models show a general aversion to entering other rooms, especially if they have encountered objects that are semantically similar to task objects. Figure 5 (bottom) shows average *epistemic gain* per turn. Like the earlier observation, higher-performing models make more effective *epistemic* actions. Thus, we can confirm that such exploration behaviour of models results
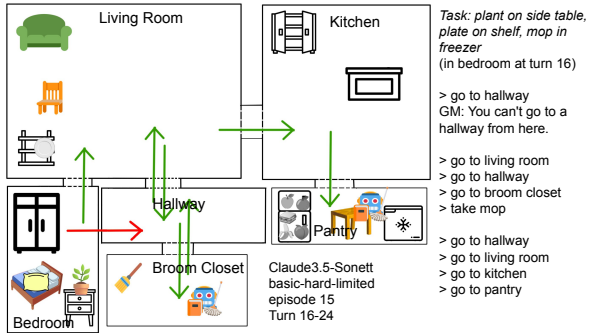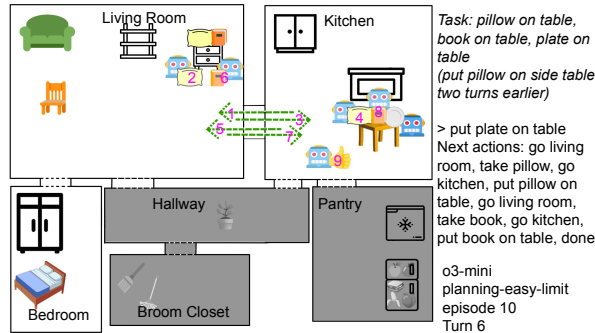
Figure 6: *Claude-3.5* correcting navigation.
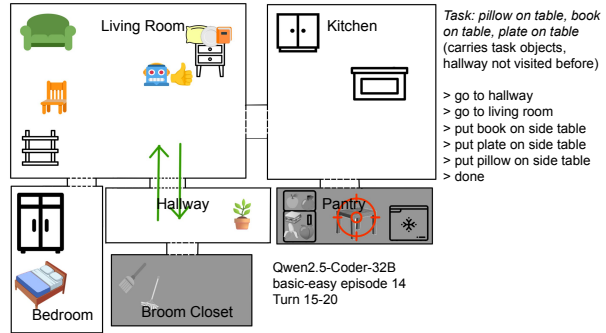

Figure 8: *Qwen2.5-32B* exploring insufficiently.

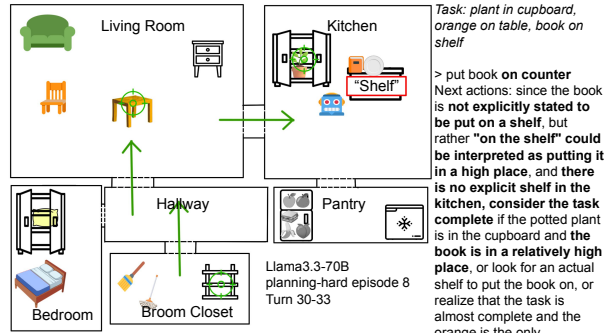Figure 7: *o3-mini* planning nine turns ahead.

Figure 9: *Llama3.3-70B* hallucinating.

in higher scores in both discovering other rooms and relevant objects in them.

5.3 Qualitative Analysis

5.3.1 Navigation

All models make navigation errors. These errors occur despite the observation feedback mentioning passages to all connected rooms every time the player enters a room. While lower-performing models repeat this type of failure many times in individual episodes, better-performing models acquire the game's rule that allows movement only to connected rooms and self-correct their navigation after receiving feedback. This type of failure most often occurs in the turn after a task object has been picked up or delivered.

Figure 6 illustrates this: *Claude3.5* attempts to "> go to hallway" from the unconnected bedroom (red arrow) and is told that this is not possible. It then respects the connection requirement for the rest of the episode and goes to the broom closet to take the mop and further to the pantry with the freezer without attempting to go to these known rooms directly.

5.3.2 Planning & Self-correction

Abstract "next actions" aid better-performing models in adequately exploring the environment and

navigating by keeping room layout and task requirements in recent context. Once sufficient information about the game situation is acquired, leading models produce extended, fully executable plans demonstrating good planning ability.

Figure 7 illustrates *o3-mini* making the longest viable plan: After finding the pillow and placing it on the first observed side table in the living room, it enters the kitchen and observes the remaining task objects. Next, it puts the plate on the table. It correctly plans nine turns ahead, self-correcting placement of pillow and book: Moving between the living room and kitchen twice, taking a task object each time and placing it on the table, preventing inventory limit infractions, and finishing the episode.

5.3.3 Exploration

Figure 8 illustrates insufficient exploration: *Qwen2.5-Coder-32B* regularly checks connected rooms, the hallway in this case, which largely contributes to its performance. However, it does not do this thoroughly enough, missing the table in the pantry and incorrectly placing the task objects on the side table, as most models do.

5.3.4 Hallucination

Hallucinations are common, with all models attempting to interact with objects that are not men-

tioned to be present and equating semantically similar objects. These hallucinations often persist over entire episodes once established, regardless of later observations contradicting them. A more concerning type of hallucination occurs in larger models that reason in their plans: The models reason (in often somewhat incomprehensible fashion) about the identity of objects, coming to wrong conclusions which they then follow, disregarding observations.

Figure 9 illustrates this with *Llama3.3-70B*: After sufficiently exploring, *finding and examining the shelf while carrying the book*, the model goes to the kitchen and puts the book *on the first observed counter*. It then produces post-hoc reasoning for this action, contradicting the task, providing a haphazard interpretation, using incoherent 'evidence' and asserting that the task is fulfilled.

6 Discussion

Compared to prior works and classic IF games, the presented environment and task are deliberately straightforward. While we do not provide human baseline data for our experiments, we are sure that this environment is trivial for humans to interact with, especially with IF familiarity. Despite the tested models showing this familiarity by producing classic IF actions with minimal prompting, they struggle with even the simple underlying common rules of this game, suggesting that if they emulate human capabilities at work when playing *AdventureGame*, they do so only in rudimentary fashion.

Behaviour differs not only between models but also between episodes with the same model, suggesting an impact of the textual surface form of the interaction. This is likely an effect of model architecture and foundation training paradigms relying solely on next-token prediction. More sophisticated behaviour hinting at underlying capabilities emerges with larger models, with training data amount and content playing the most prominent role. Structured training data like code and reinforcement learning towards aligning with instructions are reflected in model behaviour, as can be seen with *Qwen2.5-Coder-32B* strictly following the order of delivery given in the initial prompt.

Specifically, the highest-scoring models show pragmatic reasoning capabilities and can sufficiently model situations to infer the best course of action over long-turn sequences. The unexpectedly surfaced reasoning in "Next actions" shows that models can often communicate intermediate task

demands and requirements, suggesting abstraction on multiple levels. However, we find that current LLMs' reasoning can be detrimental to embodied interaction, as the trained theoretical reasoning requires no grounding and can lead to models disregarding actual observations that should instead ground model behaviour. Relying solely on language without external grounding may be detrimental to physical or otherwise embodied agents. Recently popular embedded LLM agents aim to mitigate this, but they fall short in 'grounding' the interaction by feeding language tokens into transformer LLMs instead of extending inputs to actual embodied feedback data. Certain failures might result from multi-turn conversation training and natural language feedback: Models break format to explain things to the 'user' in response (e.g., "I don't know how to interpret this 'look' action.") and attempts to move directly to a known room might be based on seemingly established conversational common ground. Thus, good emulation of conversational pragmatics can lead to worse performance.

7 Conclusion

AdventureGame performance increases with model size, progressing from generally bad situation modelling in smaller models to a middle ground of good situation modelling but frequent interaction failures, to only a pair of SOTA models fulfilling the given task in more than two thirds of cases.

Our experiments reveal the practical reasoning capabilities of current LLMs, with their leverage of common-sense world knowledge supporting embodied interaction in a text-based home environment. Tracking of the game world state interaction, agent observations and fine-grained failures show that while the required capabilities emerge in current LLMs, there are individual strengths and weaknesses. Reasoning more strongly grounded in embodied feedback may improve model interaction in the future. We look forward to building upon the current *AdventureGame* experiments with future examinations of in-context learning and pragmatic language capabilities.

Limitations

The current study is restricted to only English in its current state. While we have yet to do this, translating the prompts and adapting the underlying grammar entries is possible for other languages,

too.

The performance we measured here may not transfer to other modalities with more sophisticated demands, like visually or physically embodied agents or robots. Shridhar et al. (2021) found that while training in text-only environments is faster and less resource-intensive than training in the AI2Thor framework, agents trained in text-only environments struggled to adapt to the requirements of more complex embodiment properly.

Ethics Statement

In academic research, using paid proprietary APIs with underlying models about which little is known (training data, model architecture) is less than ideal. Currently, the models benchmarked here are the high-performing ones that are commercially used. We hope that more open models with high performance will be released soon and that proper research can be done on them.

References

Shreyas Basavatia, Keerthiram Murugesan, and Shivam Ratnakar. 2024. Starling: Self-supervised training of text-based reinforcement learning agent with large language models. *Preprint*, arXiv:2406.05872.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. Clembench: Using game play to evaluate chat-optimized language models as conversational agents. *Preprint*, arXiv:2305.13455.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. In *Computer Games*, pages 41–75, Cham. Springer International Publishing.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, and et al. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Maria Fox and Derek Long. 2003. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2017. Multi-shot ASP solving with clingo. *CoRR*, abs/1705.09811.

Luca Gioacchini, Giuseppe Siracusano, Davide Sanvito, Kiril Gashteovski, David Friede, Roberto Bifulco, and Carolin Lawrence. 2024. Agentquest: A modular benchmark framework to measure progress and improve llm agents. *arXiv preprint arXiv:2404.06411*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, and et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. 2024. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. *Preprint*, arXiv:2406.06769.

Peter A Jansen. 2021. A systematic survey of text worlds as embodied natural language environments. *arXiv preprint arXiv:2107.04132*.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134.

David Kirsh and Paul Maglio. 1994. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18(4):513–549.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. 2022. Ai2-thor: An interactive 3d environment for visual ai. *Preprint*, arXiv:1712.05474.

Lark parser. 2024. [link].

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2025. Embodied agent interface: Benchmarking llms for embodied decision making. *Preprint*, arXiv:2410.07166.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

NovaSky-Team. 2025. Sky-t1: Fully open-source reasoning model with o1-preview performance in $450 budget.

Davide Paglieri, Bartłomiej Cupiał, Sam Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. 2024. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, and et al. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2022. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.

Qinyue Tan, Ashkan Kazemi, and Rada Mihalcea. 2023. Text-based games as a challenging benchmark for large language models.

Alexey Tikhonov. 2024. Plugh: A benchmark for spatial understanding and reasoning in large language models. *arXiv preprint arXiv:2408.04648*.

Chen Feng Tsai, Xiaochen Zhou, Sierra S Liu, Jing Li, Mo Yu, and Hongyuan Mei. 2023. Can large language models play text games well? current state-of-the-art and open questions. *arXiv preprint arXiv:2304.02868*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models - a critical investigation. In *Advances in Neural Information Processing Systems*, volume 36, pages 75993–76005. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

R. Jay Wallace and Benjamin Kiesewetter. 2024. Practical Reason. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Fall 2024 edition. Metaphysics Research Lab, Stanford University.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. 2023. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, and et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. *Preprint*, arXiv:2307.13854.

8 Result details

Figure 10 shows percentages of failures by processing phase. Claude3.5 and Llama-3.1-405B produce more successful actions than the higher-scoring o3-mini.

Figure 11 shows percentages of entity-related failures. o3-mini and GPT-4o do not have inventory limit failures, while Qwen2.5-32b and Qwen2.5-72b have low amounts.

Figure 12 shows fluctuating average plan viability for a selected set of models.

Table 2 shows individual experiment performance for selected well-performing models.

Model	Experiment	clemscore	Quality	% Played	% Lose	Goal Rate	Plan Viability
o3-mini	basic-easy	76.2	81.2	93.8	12.5	85.4	0.0
	basic-easy-invlimit	76.2	81.2	93.8	12.5	81.2	0.0
	basic-hard	71.1	81.2	87.5	6.2	87.5	0.0
	basic-hard-invlimit	87.9	93.8	93.8	0.0	93.8	0.0
	planning-easy	55.9	68.8	81.2	12.5	68.8	20.9
	planning-easy-invlimit	50.8	62.5	81.2	18.8	62.5	32.8
	planning-hard	37.5	50.0	75.0	25.0	72.9	6.2
	planning-hard-invlimit	55.9	68.8	81.2	12.5	85.4	13.5
Claude-3.5	basic-easy	75.0	75.0	100.0	25.0	75.0	0.0
	basic-easy-invlimit	68.8	68.8	100.0	31.2	68.8	0.0
	basic-hard	70.3	75.0	93.8	18.8	91.7	0.0
	basic-hard-invlimit	58.6	62.5	93.8	31.2	85.4	0.0
	planning-easy	58.6	62.5	93.8	31.2	62.5	16.8
	planning-easy-invlimit	68.8	68.8	100.0	31.2	68.8	16.4
	planning-hard	60.2	68.8	87.5	18.8	85.4	9.2
	planning-hard-invlimit	42.2	56.2	75.0	25.0	72.9	5.7
GPT-4o	basic-easy	68.8	68.8	100.0	31.2	70.8	0.0
	basic-easy-invlimit	75.0	75.0	100.0	25.0	77.1	0.0
	basic-hard	41.0	43.8	93.8	50.0	77.1	0.0
	basic-hard-invlimit	28.1	37.5	75.0	43.8	75.0	0.0
	planning-easy	62.5	62.5	100.0	37.5	62.5	22.6
	planning-easy-invlimit	62.5	62.5	100.0	37.5	62.5	24.1
	planning-hard	35.2	37.5	93.8	56.2	72.9	11.9
	planning-hard-invlimit	35.2	37.5	93.8	56.2	75.0	21.2
Llama-3.1-70B	basic-easy	50.0	50.0	100.0	50.0	50.0	0.0
	basic-easy-invlimit	52.7	56.2	93.8	37.5	56.2	0.0
	basic-hard	35.2	37.5	93.8	56.2	68.8	0.0
	basic-hard-invlimit	32.8	37.5	87.5	50.0	68.8	0.0
	planning-easy	62.5	62.5	100.0	37.5	62.5	41.1
	planning-easy-invlimit	56.2	56.2	100.0	43.8	56.2	52.0
	planning-hard	16.4	18.8	87.5	68.8	68.8	14.4
	planning-hard-invlimit	29.3	31.2	93.8	62.5	70.8	24.0
Qwen2.5-32B	basic-easy	70.3	75.0	93.8	18.8	79.2	0.0
	basic-easy-invlimit	54.7	62.5	87.5	25.0	72.9	0.0
	basic-hard	6.2	12.5	50.0	37.5	52.1	0.0
	basic-hard-invlimit	1.6	6.2	25.0	18.8	39.6	0.0
	planning-easy	37.5	50.0	75.0	25.0	60.4	23.2
	planning-easy-invlimit	37.5	50.0	75.0	25.0	64.6	29.7
	planning-hard	12.5	25.0	50.0	31.2	52.1	10.4
	planning-hard-invlimit	3.9	12.5	31.2	31.2	39.6	11.6

Table 2: Scores by variant experiment for selected well-performing models. % Lose is the percentage of episodes that were finished without fulfilling all three task goals. *basic* refers to the *home delivery task*.
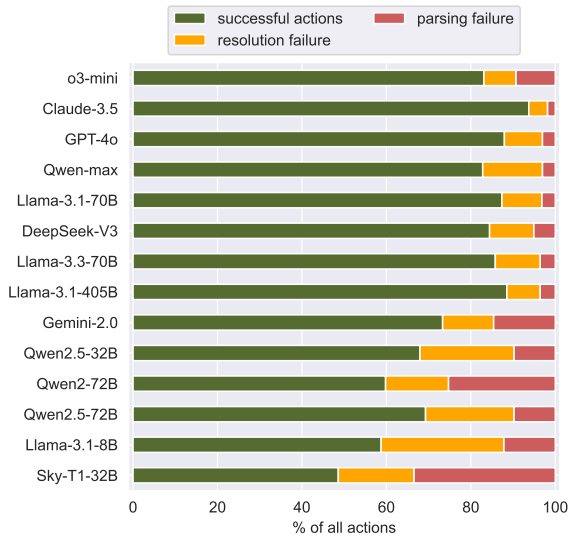
Figure 10: Successful and failed action quotas by IF interpreter processing phase.

Figure 11: Percentages of selected entity-related failures for all tested models.

9 Actions

Table 3 lists the actions defined for the experiments.

10 Initial Prompts

Prompt template for 'basic' variant instances:

```
You are playing a text adventure game. I
will describe what you can perceive in
the game. You write the single action you
want to take in the game starting with >.
Only reply with an action.
For example:
> open cupboard

Your goal for this game is: $GOAL$

Once you have achieved your goal, write
"> done" to end the game.

$INITIAL ROOM DESCRIPTION$
```

Prompt template for 'basic' with limited inventory variant instances:

```
You are playing a text adventure game. I
will describe what you can perceive in
the game. You write the single action you
want to take in the game starting with >.
Only reply with an action.
For example:
> open cupboard

You can have up to two objects in your
```
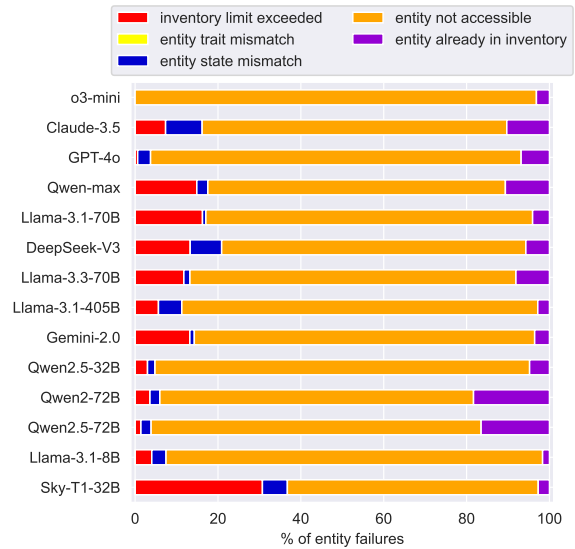
inventory at the same time.

```
Your goal for this game is: $GOAL$

Once you have achieved your goal, write
"> done" to end the game.

$INITIAL ROOM DESCRIPTION$
```

Prompt template for 'planning' variant instances:

```
You are playing a text adventure game. I
will describe what you can perceive in
the game. You write the single action you
want to take in the game starting with >.
Write your plan for your next actions on
the line below your action, starting with
"Next actions:".
For example:
> open cupboard
Next actions: take orange, eat orange

Your goal for this game is: $GOAL$

Once you have achieved your goal, write
"> done" to end the game.

$INITIAL ROOM DESCRIPTION$
```

Prompt template for 'planning' with limited inventory variant instances:

```
You are playing a text adventure game. I
will describe what you can perceive in
the game. You write the single action you
```
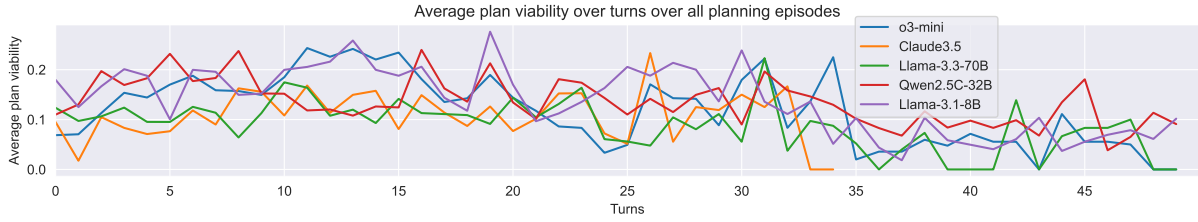
Figure 12: Average plan viability over turns over all episodes.

Action	Targets	Description	Epistemic	Pragmatic
open	'container' entities	Changes state of `closed` container entity to `open`	Yes	Yes
close	'container' entities	Changes state of `open` container entity to `closed`	No	Yes
take	'takeable' entities	Removes `in/on` state for 'takeable' entity and adds `in(entity,inventory)` fact	No	Yes
put	'takeable'& 'container'/'support' entities	Removes `in(entity,inventory)` state for 'takeable' entity and adds `in/on(entity,target)` fact	No	Yes
go	'room'	Changes at state of player entity and all entities in inventory to target room	Yes	Yes
done	-	Ends the episode	No	Yes
examine	entities	Results in entity state feedback	Yes	No

Table 3: Action types used in AdentureGame. Targets are those for which the world state holds a fact assigning the listed state.

```
want to take in the game starting with >.
Write your plan for your next actions on
the line below your action, starting with
"Next actions:".
For example:
> open cupboard
Next actions: take orange, eat orange

You can have up to two objects in your
inventory at the same time.

Your goal for this game is: $GOAL$

Once you have achieved your goal, write
"> done" to end the game.

$INITIAL ROOM DESCRIPTION$
```

The placeholder `$GOAL$` in the templates is replaced with the task goal.

Example 'easy' difficulty task goal: `Put the pillow on the table, the book on the table and the plate on the table.`

Example 'hard' difficulty task goal: `Put the pillow on the counter, the book on the shelf and the plate on the table.`

A description of the room the player is in initially is inserted at the end of the template, at the location of `$INITIAL ROOM DESCRIPTION$`.

11 Environment Graphs

To illustrate differences between 'easy'/'hard' environment and task complexity, Figures 13 and 14 show graph representations of initial game world states and task targets. House-shaped nodes are rooms, with arrow edges showing bidirectional connections between them. Round nodes are 'movable' entities, connected to rectangular receptacles and rooms by edges labelled with their prepositional state. Dashed edges connect the movable task objects to their target receptacles and are labelled with the target prepositional state.
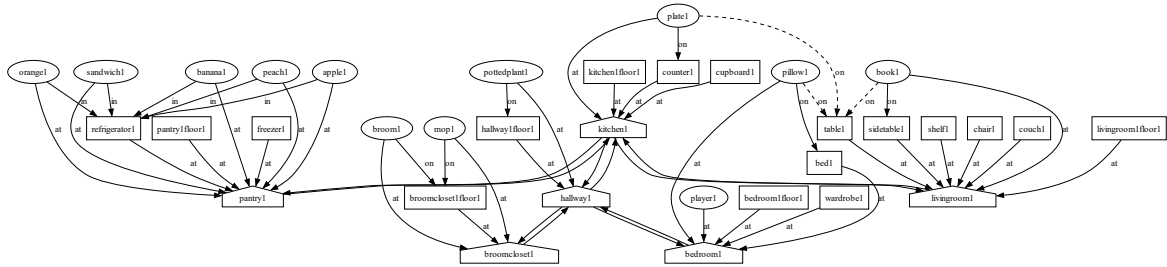
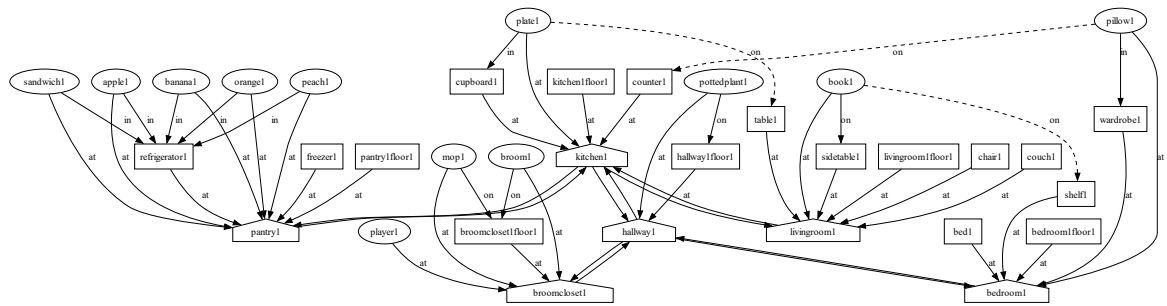Figure 13: Graph representation of an 'easy' instance.

Figure 14: Graph representation of a 'hard' instance.