The Dissertation Committee for Vanya Cohen
certifies that this is the approved version of the following dissertation:

# Reasoning about Actions with Large Multimodal Models

**Committee**:

Raymond J. Mooney, Supervisor

David Harwath

Elias Stengel-Eskin

Jacob Andreas

# Reasoning about Actions with Large Multimodal Models

**by**

**Vanya Cohen**

## Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

## Doctor of Philosophy

**The University of Texas at Austin**

**August 2026**

# Acknowledgments

To my advisor Ray Mooney, thank you for a wonderful five years as your student. I have learned so much from my time working with you. The depth and continual expansion of your knowledge, about both artificial intelligence and the world outside it, are an inspiration to me. Of all the skills I saw you demonstrate, I most appreciate having seen your highly developed ability to, in research matters or any situation, figure out the highest-order bit and keep focused on it.

I also want to thank Stefanie Tellex, Benjamin Rosman, Nakul Gopalan, and Niranjan Balasubramanian for their mentorship and for facilitating collaborations during my PhD, as well as my collaborators: David Paulius, David Watkins, Eric Rosen, Geraud Nangue Tasse, Ifrah Idrees, Jason Xinyu Liu, Matthew Gombolay, Nathanael Chambers, Shreyas Sundara Raman, Steven James, and Yash Kumar Lal. Thank you to my committee, David Harwath, Elias Stengel-Eskin, and Jacob Andreas, for their time and insights that shaped this work, and to my fellow students Albert Yu, Jierui Li, Jordan Voas, Prasoon Goyal, and Jialin Wu for all that I learned from them.

During my time as an undergraduate and master's student at Brown University, I was fortunate to have a number of mentors and collaborators who guided my path, including my research advisors Stefanie Tellex and George Konidaris, my graduate student mentor Nakul Gopalan, as well as Ellie Pavlick, James Tompkin, Benjamin Burchfiel, Thao Nguyen, Aaron Gokaslan, and Roma Patel. I am also grateful to James Kellner, who early in my college career generously took the time to encourage my interest in artificial intelligence and pursuing graduate study.

I am grateful for my family and partner Tim Genovese, who supported me throughout

my PhD. I am thankful for my mom Helena Cohen, who taught me how to use my first computer at the age of five and who has been a constant support and resource through every stage of my education, and my dad Peter Cohen Jr., who taught me how to program and to seek out interesting technical problems, lessons that stayed with me throughout this work. Together they instilled in me the value of education. I would also like to acknowledge my grandfather Peter Cohen Sr., who has been there for me throughout my life, my grandfather Johannes Schonken, who encouraged my interest in science from a young age, and my great-grandfather Joel Cohen, who in the time we had together taught me the joy and importance of being a lifelong learner.

# Reasoning about Actions with Large Multimodal Models

Vanya Cohen
The University of Texas at Austin, 2026

SUPERVISOR: Raymond J. Mooney

Large multimodal models have become central for solving sequential decision-making tasks, enabling improved learning in diverse areas such as home robotics and automated software development. However, leveraging these models for sequential decision-making requires robust action reasoning capabilities, which remain a significant challenge. This dissertation improves and evaluates action reasoning in large multimodal models. First, we introduce a method to improve the parsing of instructional texts into action sequences by integrating external symbolic planners and planning domains during autoregressive language model decoding. Next, we develop a method that leverages the compositional structure of language instructions to improve generalization and sample efficiency of acquiring new tasks with reinforcement learning. We also construct a new benchmark to evaluate the understanding of dependencies between actions described in instructional texts. Finally, we evaluate the world modeling limitations of frontier models through multimodal entity state tracking. Current models struggle to reason about the effects of actions in multimodal entity state tracking tasks. We extend entity state tracking evaluations to a simulated embodied environment and derive insights for improving the entity-state reasoning abilities of language and vision-language models. Together these contributions enhance the understanding of how models reason about actions and provide insights toward their improvement for real-world sequential decision-making problems.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

The ability to reason about actions has been a longstanding challenge for building artificial intelligence systems that interact with the world (McCarthy, 1959; Sampat et al., 2022; Georgeff, 1987; Ginsberg and Smith, 1988). Action reasoning involves understanding how actions change environment states, what constraints govern their valid execution, and how actions relate to each other and the goals of sequential decision-making tasks. This capability is critical for decision-making agents, as it enables them to plan and act effectively even in unseen scenarios.

A large portion of important AI applications can be modeled using the *sequential decision-making* framework. At each step in a sequence, the agent observes the state of the environment, takes an action, and the environment transitions to a new state. The agent continues to take actions until a problem-dependent termination criterion is met. The goal of the agent is to produce a sequence of actions which maximizes some utility defined by the domain. Progress in solving these tasks has accelerated in recent years; a key driver is the integration of large multimodal models (LMMs) as pretrained representations for decision-making agents (Yang et al., 2023; Cohen et al., 2024). These pretrained models allow for more sample-efficient learning and improved generalization, and their abilities are largely the product of training on web-scale corpora spanning diverse tasks, domains, and modalities. In many sequential decision-making tasks, goals are specified using natural language commands, and so the ability to reason about actions that derive from natural language inputs is particularly important for practical decision-making agents.

In this dissertation, we seek to answer the following question: How can the action reasoning abilities of large multimodal models be improved and evaluated to advance

sequential decision making? We address this question from two complementary perspectives. First, we develop methods that *improve* action reasoning by integrating external knowledge and compositional structure into pretrained models, and by reinforcement learning reasoning training (Chapters 3, 4, and 6 respectively). Second, we develop benchmarks that rigorously *evaluate* the action reasoning abilities of frontier models, revealing both strengths and significant limitations (Chapters 5 and 6).

## 1.1 Motivation

### 1.1.1 Improving Action Reasoning

A natural approach to improving action reasoning is to leverage the vast knowledge encoded in large language models. However, using LLMs for action reasoning in sequential decision-making presents two key challenges.

The first challenge concerns *executability constraints*. When parsing natural language instructions to formal action sequences, the resulting plans must respect the preconditions and effects defined by the environment. Pretrained language models, even when prompted with in-context examples, do not inherently enforce these constraints and may generate action sequences that are semantically plausible but physically inexecutable. A model might, for instance, instruct an agent to "slice the tomato" before "picking up the knife." We address this challenge in Chapter 3 by augmenting pretrained models with external symbolic planning knowledge, enabling them to generate action sequences that are both semantically faithful to the input instructions and executable in the target environment.

The second challenge concerns *sample efficiency and compositional generalization*. When goals are specified using language, the space of possible instructions grows combinatorially with the vocabulary of concepts. Training a separate policy for each possible instruction is infeasible. We address this challenge in Chapter 4 by exploiting the shared compositional

structure of language and actions. We represent task policies as logical compositions of reusable skill components implemented as value functions, enabling zero-shot generalization to novel compositions of tasks and substantially improved sample efficiency.

### 1.1.2 Evaluating Action Reasoning

While the methods above improve action reasoning, it is equally important to understand the *limitations* of current models. Do frontier models truly understand the causal and temporal relationships between actions? Can they track how actions change the state of entities across modalities?

The first evaluation challenge concerns *action dependencies*. Actions in sequential tasks are not independent. They have preconditions that must be satisfied and effects that enable subsequent steps. Understanding these dependencies is critical for plan execution, yet there has been no work specifically focused on evaluating whether LLMs can reason about and explain dependencies between actions in instructional texts. We address this gap in Chapter 5 by developing CaT-Bench, a benchmark that evaluates causal and temporal dependency reasoning in plans. Our findings reveal that even frontier models struggle with this form of reasoning, in many cases performing no better than random chance.

The second evaluation challenge concerns *multimodal entity state tracking*. Real-world agents must reason about how actions change entity states across modalities, not just text. A robot observing a kitchen must integrate visual observations of actions (e.g., seeing a pot being placed on a stove) with textual state representations (e.g. user instructions) to maintain a coherent world model. We address this in Chapter 6 by developing MET-Bench, a multimodal benchmark that evaluates entity state tracking across text and image modalities, and by investigating whether reinforcement learning can improve these abilities.

## 1.2 Contributions and Dissertation Outline

Chapter 2 covers relevant background on sequential decision making, symbolic planning, large multimodal models, and related work on action reasoning.

Chapters 3 through 6 cover the core technical contributions of this dissertation:

- **Chapter 3**: *Using Planning to Improve Semantic Parsing of Instructional Texts. Presented at the Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Association for Computational Linguistics (ACL) 2023.*
  - We propose a method for parsing long-form instructional texts to executable formal action sequences by augmenting in-context learning with external symbolic planners and planning domain definitions.
  - We introduce a planning-augmented decoding procedure that ranks candidate parses using planning domain knowledge and corrects plans to enforce executability constraints while preserving semantic fidelity to the input instructions.
  - We demonstrate substantial improvements in executability and plan quality on two recipe semantic-parsing datasets.
- **Chapter 4**: *Compositional Instruction Following with Language Models and Reinforcement Learning. Presented at the Reinforcement Learning Conference (RLC) 2025. Published in Transactions on Machine Learning Research (TMLR) 2024.*
  - We introduce CERLLA, a method in which policy representations for language-conditioned RL tasks are composed from conjunctions, disjunctions, and negations of pretrained compositional value functions.
  - We demonstrate a novel semantic parsing method using in-context learning with feedback from environment rollouts.

  - We solve 162 unique tasks simultaneously in an augmented MiniGrid-BabyAI domain, at the time the largest collection of simultaneously learned vision-language-RL tasks, and demonstrate substantial improvements in sample efficiency over non-compositional baselines.

- **Chapter 5**: *CaT-Bench: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans. Presented at the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2024.*

  - We introduce CaT-Bench, a question-driven evaluation framework for assessing whether LLMs can reason about causal and temporal dependencies between actions in instructional texts.
  - We curate a dataset of step-pair dependency annotations for cooking recipes and develop question templates that probe different types of step-dependence relationships.
  - We show that frontier models lack the ability to robustly reason about action dependencies, and in many cases perform no better than random chance.

- **Chapter 6**: *MET-Bench: Multimodal Entity Tracking for Evaluating the Limitations of Vision-Language and Reasoning Models. Presented at the International Conference on Machine Learning (ICML) 2026.*

  - We introduce MET-Bench, a multimodal entity tracking benchmark that extends entity state tracking evaluation from text-only settings to multimodal domains with images and text.
  - We evaluate frontier vision-language models on Chess, Shell Game, and Minecraft tasks, revealing significant performance gaps, particularly when actions are

conveyed through images rather than text, and demonstrating that even strong models fail to maintain coherent entity representations under visual state updates. We show these limitations primarily stem from deficits in visual reasoning rather than perception.

- We investigate whether Group Relative Policy Optimization (GRPO) can improve entity state tracking in open-source models, and find that while GRPO improves in-domain performance, achieving robust cross-modal transfer remains an open challenge.

Finally, Chapter 7 contains future work and Chapter 8 includes concluding remarks that summarize the contributions of this dissertation.

# Chapter 2: Background and Related Work

We review background information on reinforcement learning and sequential decision making, symbolic planning, and large multimodal models. We then introduce related work on our methods for reasoning about actions.

## 2.1 Sequential Decision Making

Many important artificial intelligence problems can be formulated in the *sequential decision making* framework. At each step in a sequence, the *agent* model receives an observation representing the state of the *environment* and decides on what action to take. The action affects a *transition* in the state of the environment and the cycle repeats until a problem-dependent termination criterion is met. The goal of the agent is to produce a sequence of actions which maximizes some *utility* defined by the domain.

In this dissertation, we consider two standard frameworks for deciding what actions the agent should take to maximize utility and thus solve sequential decision-making tasks: symbolic task planning and reinforcement learning.

### 2.1.1 Symbolic Task Planning

A task planning domain defines an environment, actions that modify the environment, and a transition function specifying the effect of actions on the environment (Ghallab et al., 2016). The environment state is composed of a collection of Boolean values defining the existence and state of objects and planning actions are implemented as STRIPS-style operators (Fikes and Nilsson, 1971). Each action has logical preconditions that must be satisfied for its execution. An action is said to be *executable* if its preconditions are met.

For any state, the afforded or executable actions are all actions with satisfied preconditions. Upon execution the action changes the values of the variables which define the environment. A planning task is specified by an initial state and goal conditions, in a planning language like PDDL (Fox and Long, 2003). The resulting plan (if it exists) comprises a sequence of actions that can be taken to reach a goal state satisfying those conditions. These tasks are solved by a *planner* which searches for this sequence of actions using implementations like Fast Forward and Fast Downward (Hoffmann and Nebel, 2001; Helmert, 2006).

### 2.1.2 Reinforcement Learning (RL)

In reinforcement learning, the agent receives a learning signal from the environment in the form of a scalar reward at each step. It must learn to maximize the utility represented by the temporally-discounted cumulative reward over all steps (Sutton and Barto, 2018). RL formalizes the environment as a Markov decision process (MDP) represented as a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is the state space of the environment and $\mathcal{A}$ are the actions available to the agent. The transition function $p(s'|s, a)$ specifies the probability of the environment entering state $s'$ after action $a$ in state $s$. The learning signal is provided by the reward function $r(s, a)$, which is the reward for executing $a$ in $s$. $\gamma \in [0, 1)$ is the temporal discount factor, which ensures the infinite-horizon sum of rewards remains finite.

**Acting in an MDP:** There are two primary approaches to solving an MDP. The first is to learn an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that directly maps states to actions. The optimal policy $\pi^*$ is that which obtains the greatest expected return at each state: $V^{\pi^*}(s) = \max_\pi V^\pi(s)$ for all $s \in \mathcal{S}$, where the value function of policy $\pi$ is given by $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \, r(s_t, a_t)\right]$ and represents the expected return after executing $\pi$ from $s$. The second approach is to learn an optimal value function and derive a policy from these value estimates. Closely related to

the state-value function is the action-value function, $Q^\pi(s, a)$, which represents the expected return obtained by executing $a$ from $s$, and thereafter following $\pi$. The optimal action-value function is given by $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. Given the optimal action-value function, an optimal policy can be recovered by acting greedily with respect to it: $\pi^*(s) = \arg\max_a Q^*(s, a)$.

### 2.1.3 Compositional Action Representations

Many tasks have underlying compositional structure: each task can be represented by its component subtasks and how these subtasks are combined (Szabó, 2020). Learning representations which match this compositional structure enables better generalization and more sample efficient learning (Andreas et al., 2016; Hu et al., 2018). In MDPs, high-level actions can be implemented as low-level policies (Sutton et al., 1999). These low-level policies can be implemented as compositional value functions (Todorov, 2007; Van Niekerk et al., 2019; Haarnoja et al., 2018; Hunt et al., 2019) which enable the representation of high-level actions as the composition of low-level value functions. Nangue Tasse et al. (2020) extend these to provably zero-shot optimal compositions for logical operators in stochastic shortest path problems. BabyAI (Chevalier-Boisvert et al., 2019) provides a large number of language-RL tasks suitable for evaluating compositional action representations. Another compositional RL benchmark, CompoSuite (Mendez et al., 2022), does not include language and has fewer unique goal conditions. Kuo et al. (2021) demonstrate compositional representations for policies, but depend on a pre-trained parser and demonstrations.

Recent works like *SayCan* use language models and pretrained language-conditioned value functions to solve language-specified tasks using few-shot and zero-shot learning (Ahn et al., 2023). Shridhar et al. (2021a) use pretrained image-text representations to perform robotic pick-and-place tasks. Other work incorporates learning from demonstration and

language with large-scale pretraining to solve robotics tasks (Driess et al., 2023; Brohan et al., 2022). However, these works use learning from demonstration as opposed to RL, do not support negations of pre-trained value functions, and are unsuitable for continual learning settings where both RL value functions and language embeddings are improved over time. Schick et al. (2023) use an LLM to learn a semantic parser in a weakly supervised setting.

### 2.1.4 RL for Reasoning

Recent work demonstrates the effectiveness of reinforcement learning for improving reasoning capabilities in language models (Lambert et al., 2024). DeepSeek-R1 (DeepSeek-AI et al., 2025) shows that RL training with verifiable rewards can significantly enhance mathematical and logical reasoning with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). This policy-gradient method samples a group of rollouts for the same prompt and computes each rollout's advantage relative to the group's reward distribution, avoiding the need for a separate value model. We apply GRPO to multimodal entity tracking in Chapter 6.

## 2.2 Large Multimodal Models (LMMs)

Many recent advances in artificial intelligence have been enabled by training transformer-based models with large parameter counts (Vaswani et al., 2017) on data scraped from the internet (Radford et al., 2018; Devlin et al., 2019). These models learn general purpose representations that are capable of solving diverse tasks across domains, even for domains and tasks that are not explicitly represented in the training data (Radford et al., 2019). Recent work has extended these text-based models to multimodal data which may combine images, text, audio, video, and actions from sequential decision-making environments,

among other modalities (Li et al., 2023a).

### 2.2.1 In-Context Learning

In-context learning allows pretrained models to perform tasks with a small number of input-output pairs (Brown et al., 2020b). These examples are provided to condition the language model to produce more performant outputs. Typically these examples are drawn from the training split of a dataset and prepended to a test example to condition further autoregressive generation.

## 2.3 Action Reasoning

Reasoning about actions is fundamental to artificial intelligence systems that interact with and transform their environments. This involves understanding how actions change environment states, what constraints govern their valid execution, and how actions relate to each other and goals in sequential decision-making tasks. We focus on techniques which utilize large multimodal models to implement these types of reasoning.

### 2.3.1 Mapping Language Instructions to Actions

There are a variety of techniques for learning a mapping from instructional texts to symbolic action representations suitable for sequential decision-making tasks. These include supervised machine translation (Wong and Mooney, 2006), reinforcement learning-based methods (Branavan et al., 2009, 2010), deep-learning methods (Papadopoulos et al., 2022), and recent few-shot semantic parsing (Shin et al., 2021; Shin and Van Durme, 2021; Chen et al., 2021), among others (Cohen et al., 2024). Other recent work uses few-shot learning and large language models to map single commands (Huang et al., 2022; Anderson et al., 2018; Blukis et al., 2020; Chaplot et al., 2018) and short sequences of commands (Ahn et al.,

2023) to actions. However, these works lack explicit compositionality and thus struggle to generalize to novel commands. These methods also lack guarantees about the executability of generated action sequences. Other approaches leverage compositionality by mapping to a compositional symbolic representation and then planning over the symbols (Dzifcak et al., 2009; Williams et al., 2018; Gopalan et al., 2018); however, these methods do not learn the semantics of these symbols. Our work goes beyond these limitations by incorporating external symbolic planning domains into the instruction-action mapping function to simultaneously optimize the executability, and separately, by leveraging compositional high-level actions to enable systematic generalization to novel instructions.

### 2.3.2 Understanding Action Dependencies and Effects

Early work in text understanding argued for the importance of understanding plans and goals (Schank and Abelson, 1977a). Generating plans involves different types of understanding such as temporal reasoning and entity state tracking. NaturalPlan (Zheng et al., 2024) presents real-world tasks with natural language interaction, but is limited to three tasks. PlanBench (Valmeekam et al., 2023) showed that LLMs were unable to generate executable and effective plans, but focused on simulated worlds with restrictive PDDL syntax. Prior work proposed the Goal-Oriented Script Construction task, where a model produces a sequence of steps (or a plan) to accomplish a given goal. ChattyChef (Le et al., 2023) uses the conversational setting to generate cooking instructions and iteratively refine its step ordering. CoPlan (Brahman et al., 2023) collects conditions associated with a revised list of steps for the task of plan revision to satisfy constraints. Other work studies the use of LLMs for plan customization according to user requirements. LLMs have been shown to generate plans well but it is unclear how well they truly understand all aspects of these plans.

Plan understanding tasks involve multiple aspects such as tracking entity states

(Bosselut et al., 2018; Henaff et al., 2017), linking actions (Pareti et al., 2014), next event prediction, and more. OpenPI (Tandon et al., 2020) enables entity tracking in how-to procedures. ProPara (Dalvi et al., 2018) focuses on describing and understanding scientific processes. XPAD (Dalvi et al., 2019) extends ProPara by adding the task of explaining actions by predicting their dependencies. Other work formalizes multiple-choice tasks related to step- and goal-relations in procedures. Kiddon et al. (2015) explore predicting dependencies in cooking recipes and related tasks. Similar work has been done on identifying dependencies in multimodal instructions with images and text (Pan et al., 2020; Wu et al., 2024). PizzaCommonsense (Diallo et al., 2024) is a dataset for learning commonsense about intermediate and implicit steps for cooking recipes, and contains explicit input/output pairs of each action with fine-grained annotations. Choice-75 (Hou et al., 2023) aims to study decision branching in plans by generating user scenarios and choices for various steps within the plan. CREPE (Zhang et al., 2023) measures how well LLMs understand the comparative likelihood of two events occurring in a procedure. There are a variety of datasets evaluating different aspects of plans, but there is a lack of one that clearly studies the prediction and explanation of temporal ordering constraints on the steps of an instructional plan.

### 2.3.3 Multimodal Entity Tracking

Various works have directly and indirectly studied entity state tracking, including in multimodal settings. MET-Bench differs from these by studying parallel multimodal entity state tracking tasks in text and image, allowing for analysis of the unique challenges of tracking entities over analogous tasks in both modalities.

**Textual State Tracking** Entity tracking has been extensively studied in textual domains, with a focus on probing and improving LLMs' abilities to maintain representations of entity

states. Toshniwal et al. (2022) evaluate chess as an entity tracking domain, employing finetuned models (Radford et al., 2019) to assess performance. Kim and Schuster (2023) examine the impact of model size and finetuning on entity tracking in textual settings similar to our Shell Game domain. Tandon et al. (2020) construct a benchmark for understanding entity state changes in procedural texts.

Several studies explore the implicit representations of entity states in LLMs. Li et al. (2025a) employ interpretability methods to understand how LLMs implement entity tracking. Li et al. (2021) and Long et al. (2016) use semantic probing to reveal that Transformer-based models (Vaswani et al., 2017) capture entity state representations implicitly during textual reasoning. Building on this, Prakash et al. (2024) demonstrate that finetuning language models for entity tracking tasks enhances pre-existing internal mechanisms rather than learning entirely new representations. Li et al. (2023b) find that Transformers trained on Othello games form internal representations of the game state.

Efforts to improve textual entity tracking beyond domain-specific finetuning include Fagnou et al. (2024), which establishes theoretical limitations of the Transformer architecture in tracking entities. They propose a novel attention mechanism for entity tracking in Transformers. Gupta and Durrett (2019) train small Transformer-based models for tracking entity state in instructional texts. Kim et al. (2024) investigate how code pretraining improves language models' abilities to track entities in text, while Yoneda et al. (2024) introduce Statler, a prompting method designed to maintain accurate state representations in text-based robotics planning. Concurrent work uses RL for text-state prediction for single-step transitions (Yu et al., 2026); our work uses RL to train multi-step entity state tracking from images and text.

**Multimodal Procedure Understanding** Multimodal procedure understanding evaluates the ability of models to properly sequence images depicting recipe states (Shirai et al., 2022)

or to localize or describe object states (Nguyen et al., 2024; Xue et al., 2024). In contrast, MET-Bench requires tracking state through sequential updates from actions specified in parallel text and image modalities.

**World Modeling Evaluation** World modeling evaluations focus on video generation (He et al., 2025; Li et al., 2025b) and prediction tasks (Gao et al., 2025; Bordes et al., 2025) that lack the combination of parallel modalities, sequential tracking, and scalable difficulty in MET-Bench.

**Tracking and Segmentation** Natural language and vision tracking and segmentation benchmarks evaluate spatial acuity in bounding box placement and pixel-level segmentation of objects and entities (Wang et al., 2021; Feng et al., 2023; Carion et al., 2025). MET-Bench is a complementary task that requires predicting the world state after visual action sequences. The Chess and Shell Game domains involve trivial localization of the relevant entities, yet models fail to reason about sequential visual state updates.

# Chapter 3: Using Planning to Improve Semantic Parsing of Instructional Texts

*Presented at the Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Association for Computational Linguistics (ACL) 2023.*

## 3.1 Introduction

Action reasoning in sequential decision-making requires that generated action sequences respect the preconditions and effects that govern valid execution. When a pretrained language model parses natural language instructions into a formal plan, it may produce an ordering that is semantically plausible but physically inexecutable: an action whose preconditions have not yet been satisfied cannot be carried out, regardless of how fluent the parse appears. Instructional texts such as recipes make this failure mode concrete and frequent. This chapter addresses it by integrating external symbolic planning knowledge into LLM-based semantic parsing, so that generated action sequences satisfy the executability constraints of the target planning domain while remaining faithful to the input text. The reliance on an explicit planning domain distinguishes this work from the chapters that follow: Chapter 5 evaluates whether frontier models can reason about action dependencies without external planning information. In Cohen and Mooney (2023), we propose a planning-augmented semantic parsing method for mapping long-form instructional texts to formal action sequences and validate it on two recipe datasets (Bollini et al., 2013; Tasse and Smith, 2008).

## 3.2 Approach

Building on work from Shin et al. (2021), we investigate semantic parsing in the few-shot setting using OpenAI's Codex language and code LLM (Chen et al., 2021; Shin and Van Durme, 2021). Learning occurs in-context, by prompting the LLM with a few input-output task examples.

To address the highlighted challenges with long instructional-text parsing, we propose *Planning-Augmented Semantic Parsing*. Our method leverages a formal symbolic planning representation to rank and correct candidate plans. Plans are corrected by searching for sequences of actions that satisfy the preconditions of all output actions. Ranking selects plans which best meet the domain's planning and syntactic constraints by ranking plans highly if they have fewer inadmissible actions, require fewer additional actions to correct, and have fewer steps with invalid syntax. After ranking, planning errors are fixed using symbolic planning methods. The result is an effective neuro-symbolic approach that combines the strengths of deep-learned LLMs and classical AI planning.

We use Davinci Codex (Chen et al., 2021) based on the GPT-3 architecture (Brown et al., 2020b). Like GPT-3, the model was trained on a large web-sourced text corpus, but also includes code in the dataset. While OpenAI does not publish the sizes of the Codex models available through their API, Gao (2021) empirically estimates the size of text-only Davinci at 175B parameters.

### 3.2.1 Datasets

We evaluate our method on two recipe semantic parsing datasets Tasse and Smith (2008) and Bollini et al. (2013). Tasse and Smith (2008) contains 260 recipes with corresponding semantic parses in the Minimal Instruction Language for the Kitchen (MILK) syntax. Each statement in the language corresponds to a plan step with an action and

arguments. Some plan steps produce new variables (ingredients or tools) which are consumed by subsequent steps. The recipes in this dataset cover a wide range of cuisines, ingredients, techniques, and tools. Each step also contains an optional human-readable description of the step. The 60 recipes of Bollini et al. (2013) were selected to be executed by a cooking robot. The recipes are mainly limited to baking and contain a small fixed set of tools and actions. This dataset also contains planning domain definitions in the form of STRIPS-style operator actions (Fikes and Nilsson, 1971).

### 3.2.2 Planning Domains

Bollini et al. (2013) contains planning domain definitions that specify the state of the kitchen, ingredients, and tools used in each recipe. These were developed to facilitate recipe execution on a real-world cooking robot. The Tasse and Smith (2008) dataset does not provide planning definitions. We construct planning definitions where the existence of ingredients and tools are the only predicates and transitions in the environment involve either creating or destroying these objects. Actions are considered valid if their objects have been instantiated by prior steps; otherwise, their preconditions are considered unsatisfied.

| **Tasse and Smith (2008)** | |
|---|---|
| A Very Intense Fruit Smoothie | |
| 1 (10 ounce) package frozen mixed berries<br>1 (15 ounce) can sliced peaches, drained<br>2 tablespoons honey<br>In a blender, combine frozen fruit, canned fruit and honey.<br>Blend until smooth. | create_ing(“1 (10 ounce) package frozen mixed berries”)<br>create_ing(“1 (15 ounce) can sliced peaches, drained”)<br>create_ing(“2 tablespoons honey”)<br>create_tool(“blender”)<br>combine({“1 (10 ounce) package frozen mixed berries”, “1 (15 ounce) can sliced peaches, drained”, “2 tablespoons honey”}, “fruit and honey”, “”)<br>put(“fruit and honey”, “blender”)<br>mix(“fruit and honey”, “blender”, “smoothie”, “blend”)<br>chefcheck(“smoothie”, “smooth”) |
| **Bollini et al. (2013)** | |
| Easy Cake Mix Cookies | |
| 1 (18 1/4 ounce) box chocolate cake mix<br>1/3 cup vegetable oil<br>2 eggs<br>Combine cake mix, oil and eggs.<br>Mix well.<br>Bake at 350F for about 10 minutes.<br>Remove from oven and let cool on pan for several minutes before removing to rack to finish cooling. | ingredient([“cake_mix”], “1 (18 1/4 ounce) box”, homogenous=True)<br>ingredient([“oil”], “1/3 cup”, homogenous=True)<br>ingredient([“eggs”], “2”, homogenous=True)<br>pour(cake_mix), pour(oil), pour(eggs), mix()<br>mix()<br>scrape(), preheat(350), bake(10)<br>noop() |

Table 3.1: Example recipes from the (Tasse and Smith, 2008) and (Bollini et al., 2013) datasets. These examples are the second shortest and shortest for each dataset respectively.

| Bollini et al. (2013) |
|---|
| ingredient(contains : string, amount : string, homogenous : bool) |
| pour(ingredient : string) |
| scrape() |
| preheat(temperature : string) |
| bake(time : string) |
| noop() |

| Tasse and Smith (2008) |
|---|
| create_tool(name : string) |
| create_ing(name : string) |
| chefcheck(name : string, description : string) |
| cut(item : string, tool : string, result : string, description : string) |
| combine(item : string, tool : string, result : string, description : string) |
| cook(item : string, tool : string, result : string, description : string) |
| do(item : string, tool : string, result : string, description : string) |
| leave(item : string, description : string) |
| mix(item : string, tool : string, result : string, description : string) |
| put(item : string, tool : string) |
| remove(item : string, tool : string) |
| separate(item : string, result1 : string, result2 : string, description : string) |
| serve(item : string, description : string) |
| set(item : string, description : string) |

Table 3.2: Action definitions for the cooking recipe domains. The actions of Bollini et al. (2013) operate on a world-state definition that includes the state of the ingredients, a mixing bowl, a baking pan, and the oven. In contrast Tasse and Smith (2008) provides no planning domain definitions and employs qualitative descriptions of state transformations in the action annotations. We built a simple planning domain where the state consisted only of the existence of objects as state variables. The preconditions for an action were met if the items used in the action had been instantiated.

### 3.2.3 Prompt Design

To generate a plan for a test example using few-shot learning, we prompt the model with sample recipes and parses taken from the held-out examples. Following Liu et al. (2021), prompt examples are selected using nearest-neighbor search using the cosine similarity

between their embeddings as computed by a text embedding model, specifically the "all-mpnet-base-v2" model from the SentenceTransformers library (Reimers and Gurevych, 2019) based on the MPNet model (Song et al., 2020). The selection of few-shot training examples reduces to Equation 3.1, where $a$ is a training example, $X$ denotes the set of held-out examples, and $E$ represents the recipe embedding function.

$$P(a) = \arg\max_{x \in X} \{cos\_sim(E(x), E(a))\} \tag{3.1}$$

### 3.2.4 Planning-Augmented Decoding

Due to utilization limits of the Codex API, our method samples a fixed number of plan completions for each recipe. For sampling next tokens, we use nucleus sampling introduced by Holtzman et al. (2020), which offers improvements over other sampling methods. These are then ranked using a scoring function based on the generation probability and a *planning score* which factors in the number of precondition errors and syntax errors in the plan and the sequence probability of the plan.

The planning score is calculated by combining measures of plan executability: the number of precondition errors, syntax errors, and additional planning steps. Precondition errors occur when a plan step's preconditions are not satisfied. For example, when the step references non-existent ingredients or the world state does not allow the action to occur. Syntax errors occur when the plan contains malformed steps that cannot be parsed by the plan interpreter. The syntax error score ($SE$) is the number of plan steps which contain syntax violations. The precondition error score ($PE$) is the number of plan steps which cannot be executed because their preconditions are not satisfied. Finally, ($AS$) is the number of steps added to the plan by planning in order to maximize the number of plan steps with valid preconditions. Steps with errors are counted as opposed to counting all errors in each step.

This allows for computation of a score in the interval $[0, 1]$ to match the sequence probability score. In general, identifying multiple syntax errors in a given step is not possible, as the presence of even a single syntax error may result in an undefined grammatical context.

These counts are normalized by the plan length ($N$) so as not to penalize longer plans. The natural log is taken to re-scale the planning score for addition with the sequence log-probability, adding $\epsilon$ to avoid taking the logarithm of zero in cases where the plan contains no errors and requires no additional steps to be valid. The planning score is added to the mean log-probability of the token sequence representing the plan. This scoring function results in plans with a higher sequence probability and fewer planning errors being selected.

$$\text{score} = \ln\left(1.0 - \frac{SE + PE + AS}{N} + \epsilon\right) + \frac{1}{T}\sum_{t=1}^{T} \ln P_t \tag{3.2}$$

The plan that maximizes the $score$ function is passed to a planning module. For each inadmissible step where the preconditions are not satisfied, it searches for sequences of admissible actions to insert into the plan, such that those actions lead to valid preconditions for that step. To limit the search space, the planning module only searches in the space of plans that can be inserted before an existing inadmissible plan step.

### 3.2.5 Correcting Plans

While the ranking procedure ensures that high probability and low-error plans will be surfaced, these plans may still contain precondition errors. The planning domain information and a planning algorithm together form a planning module that can attempt to correct these precondition errors. The ranking procedure incorporates this planning module to calculate the additional steps $AS$ that can be inserted into a plan to fix precondition errors. These steps are also included in the total number of steps $N$. Therefore, fixable plans will receive a

**Algorithm 1:** The planning algorithm inserts steps before actions with unmet preconditions.

**Input:** A starting state $S$, desired action $A$, transition function $T$, and search depth $N$.
**Output:** The shortest sequence of actions $P$ which ends in action $A$ or $null$ if none exists.

```
sequences = successors(S, T, depth=N);
best = null;
for s in sequences do
    if A in s then
        /* truncate the plan until the action A or false if no
           such prefix exists                                    */
        plan = prefix(s, A);
        if plan && length(plan) < length(best) then
            best = plan;
        end
    end
end
```

better planning score.

Algorithm 1 describes the procedure for finding steps to insert into a plan to ensure that each step's preconditions are satisfied. As input, it takes the world state after executing some number of plan steps, and computes the sequence of actions needed to ensure the preconditions of the next plan step $A$ are met. The algorithm returns the shortest number of actions required to satisfy the step's preconditions. Aside from producing more valid plans, this method should produce plans which correspond more closely to the ground truth semantic parse annotations. However, because the planning module only inserts steps into a plan before an inadmissible step, and does not change the existing steps, it cannot necessarily fix all precondition errors in a plan. Utilizing planning and likelihood-based decoding balances the desire for plans with valid preconditions while ensuring that plans contain relevant steps to the recipe. These two requirements may compete in some cases. In the simplest case, an empty plan, there are no potential precondition errors, but the plan also

contains no relevant plan steps. In practice there exists a trade-off between plan executability and correctness of semantic parsing as noted by (Huang et al., 2022).

## 3.3 Experiments

### 3.3.1 Baselines

We evaluate baseline and ablated versions of our methods for ranking the generated plans: *Rank (Prob)*, and *Rank*. Our full ranking method with partial planning is denoted *Plan + Rank*. *Rank (Prob)* selects the plan with the lowest perplexity (Prob) (the highest sequence probability), providing a baseline where no planning domain information is utilized. *Rank* ranks the plans by the scoring function in Equation 3.2, but does not correct precondition errors through planning like *Plan + Rank*.

### 3.3.2 Metrics

**Longest Common Subsequence (LCS)** Prior work evaluates plan correctness in terms of the LCS between generated and ground truth plans (Puig et al., 2018). We normalized LCS by the length of the longer plan. LCS evaluates the textual overlap between plans; computing common subsequences which may contain interwoven unequal sequences. It does not strongly penalize erroneous injected subsequences. This metric ranges from $[0.0, 1.0]$ where $0.0$ indicates no sequence overlap and $1.0$ indicates identical plans.

**Plan Steps F1** LCS reflects the order and content of the generated plan steps compared to the ground truth. We also report an F1 measure (the harmonic mean of precision and recall) that quantifies the quality of the individual plan steps without regard to their sequencing.

**Precondition and Syntax Errors (PE & SE)** The previous metrics assess similarity to the ground truth plan and do not explicitly reflect the executability of generated plans. Therefore, we also measured the frequency of precondition and syntax errors in plans. Errors were counted on a per-step basis. If a step contained more than one error or more than one type of error these were quantified as a single error and type for the step.

### 3.3.3 Implementation

We utilize Davinci Codex[1] for all experiments due to its large context size of $8{,}000$ tokens which is sufficient for all prompts and completions across both datasets. Our method samples ten completions up to a fixed length of $1{,}500$ tokens or until the special delimiter sequence is reached. The decoding length is chosen to be longer than the longest recipe parse in either of the datasets. We generate ten completions for each recipe to offer a diversity of plans to rank and correct through planning. We also utilize a nucleus sampling top-p value of $0.5$. This value is selected because it maximizes the performance of the baseline with respect to LCS. We perform ten trials to compute means and confidence intervals.

The full prompt was formed by concatenating the training example instructions with their semantic parses, and the instructions for the test example. Each component of the prompt was separated by new line characters, and a special delimiter "###" which was also appended to the end of the prompt. Because of the in-context learning, the model learned to append the delimiter to the end of the generated parse, which was used to identify the end of the model's sequence completion.

[1]The OpenAI API name for the model is "code-davinci-002".

### 3.3.4 Results

| Models | Bollini et al. (2013) | | | |
|---|---|---|---|---|
| | LCS↑ | PE↓ | SE↓ | F1 ↑ |
| **Rank (Prob)** | | | | |
| Davinci Codex, E=1 | $0.897 \pm 0.008$ | $0.962 \pm 0.685$ | $0.042 \pm 0.008$ | $0.784 \pm 0.004$ |
| Davinci Codex, E=5 | $0.949 \pm 0.005$ | $0.198 \pm 0.009$ | $0.002 \pm 0.004$ | $0.863 \pm 0.003$ |
| **Rank** | | | | |
| Davinci Codex, E=1 | $0.901 \pm 0.008$ | $0.382 \pm 0.037$ | $0.025 \pm 0.008$ | $0.798 \pm 0.002$ |
| Davinci Codex, E=5 | $0.952 \pm 0.005$ | $0.120 \pm 0.015$ | $0.002 \pm 0.004$ | $\mathbf{0.868 \pm 0.002}$ |
| **Plan + Rank** | | | | |
| Davinci Codex, E=1 | $0.903 \pm 0.008$ | $0.143 \pm 0.033$ | $0.025 \pm 0.008$ | $0.807 \pm 0.002$ |
| Davinci Codex, E=5 | $0.952 \pm 0.005$ | $\mathbf{0.033 \pm 0.000}$ | $0.002 \pm 0.004$ | $\mathbf{0.870 \pm 0.002}$ |

Table 3.3: Results and ablations for the Bollini et al. (2013) dataset, reported as means over the leave-one-out cross validation. $95\%$ confidence intervals are computed using a t-distribution over ten trials. Results using one (E=1) and five (E=5) training examples in each prompt are shown. All plans are generated using a nucleus sampling top-p value of $0.5$.

| Models | Tasse and Smith (2008) | | | |
|---|---|---|---|---|
| | LCS↑ | PE↓ | SE↓ | F1 ↑ |
| **Rank (Prob)** | | | | |
| Davinci Codex, E=1 | $0.692 \pm 0.003$ | $0.827 \pm 0.086$ | $0.875 \pm 0.199$ | $0.443 \pm 0.002$ |
| **Rank** | | | | |
| Davinci Codex, E=1 | $0.695 \pm 0.004$ | $0.293 \pm 0.016$ | $\mathbf{0.226 \pm 0.024}$ | $0.446 \pm 0.001$ |
| **Plan + Rank** | | | | |
| Davinci Codex, E=1 | $0.695 \pm 0.003$ | $\mathbf{0.000 \pm 0.000}$ | $\mathbf{0.237 \pm 0.018}$ | $0.446 \pm 0.001$ |

Table 3.4: Results and ablations for the Tasse and Smith (2008) dataset, reported as means over the leave-one-out cross validation. $95\%$ confidence intervals are computed using a t-distribution over ten trials. Results using one (E=1) training example in each prompt are shown. All plans are generated using a nucleus sampling top-p value of $0.5$.

We report results for the Bollini et al. (2013) dataset in Table 3.3 and the Tasse and Smith (2008) dataset in Table 3.4. Mean cross-validation results are reported with $95\%$ confidence intervals computed using a t-distribution for ten trials. For both datasets, the

number of precondition errors is reduced by ranking using our scoring metric and corrective planning. The *Rank* method results in a decrease in the precondition error rate, and by adding corrective planning (*Plan* + *Rank*), the error rate is again reduced significantly. This results in more valid, executable plans. Even as the precondition error rate is reduced, the LCS remains constant for the Bollini et al. (2013) dataset and only slightly reduced for the Tasse and Smith (2008) dataset. This indicates that the plans maintain high agreement with the ground truth plans while steps are added. For the Bollini et al. (2013) dataset, the F1 score also improves through ranking and corrective planning indicating that highly ranked plans contain more accurate selections of recipe steps. The F1 score does not improve for the Tasse and Smith (2008) dataset and the *Plan* + *Rank* method results in no precondition errors. However, these results are not surprising because for this dataset the planning module can only insert ingredient and tool definitions into the plan. These inserted actions can result in lower LCS by lengthening the generated plan and may not match ingredient definitions in the ground truth plan. Because of the presence of free-text descriptions and specifications in the Tasse and Smith (2008) dataset and the difficulty of parsing longer plans, both the LCS and F1 are lower than for the Bollini et al. (2013) dataset. Finally, providing more in-context examples for the Bollini et al. (2013) dataset improves performance for all measured metrics.

### 3.3.5 Qualitative Examples

| **Bollini et al. (2013)** | | |
|---|---|---|
| **Rank** | **Plan + Rank** | **Ground Truth** |
| pour(nuts)<br>mix()<br>scrape()<br>bake(25) | pour(nuts)<br>mix()<br>scrape()<br>**preheat(350)**<br>bake(25) | pour(nuts)<br>mix()<br>scrape()<br>**preheat(350)**<br>bake(25) |
| **Tasse and Smith (2008)** | | |
| **Rank** | **Plan + Rank** | **Ground Truth** |
| combine("1/2 cup all-purpose flour", "1 egg", "1 tablespoon chopped fresh parsley", "1/2 teaspoon salt", "1/4 teaspoon freshly ground nutmeg", "mashed potatoes", "dough", "stir in") | **create_ing("1/2 cup all-purpose flour")**<br>...<br>combine(**"1/2 cup all-purpose flour"**, "1 egg", "1 tablespoon chopped fresh parsley", "1/2 teaspoon salt", "1/4 teaspoon freshly ground nutmeg", "mashed potatoes", "dough", "stir in") | **create_ing("1/2 cup all-purpose flour")**<br>...<br>combine(**"1/2 cup all-purpose flour"**, "1 egg", "1 tablespoon chopped fresh parsley", "1/2 teaspoon salt", "1/4 teaspoon freshly ground nutmeg", "mashed potatoes", "mashed potato mixture", "stir in") |

Table 3.5: Excerpted examples of improved parsing for recipes using the *Plan + Rank* method. The parses were selected by randomly selecting a recipe where the *Plan + Rank* method resulted in an improvement in the number of precondition errors over the baseline methods. NO-OP actions are omitted for brevity.

Table 3.5 contains qualitative examples of the *Plan + Rank* performance. For the example from the Bollini et al. (2013) dataset, in the generated plan the oven was not preheated before baking. The corrected plan added a preheat() action to satisfy the preconditions of the bake() action, which requires a heated oven. In the example from the Tasse and Smith (2008) dataset, a recipe that uses certain ingredients to make dough. In the generated plan these ingredients were not instantiated. However, the planning module

inserted actions to instantiate the ingredients which improved the validity of the generated plan. An additional example for the Bollini et al. (2013) dataset is included in Table 3.6 showing the addition of the scrape action which transfers the mixture to the baking sheet before baking.

| **Bollini et al. (2013)** | | |
|---|---|---|
| **Rank** | **Plan + Rank** | **Ground Truth** |
| pour(flour) | pour(flour) | pour(flour) |
| mix() | mix() | mix() |
| mix() | mix() | **scrape()** |
| preheat(350) | preheat(350) | preheat(350) |
| bake(20) | **scrape()** | bake(20) |
| | bake(20) | |

Table 3.6: Additional excerpted generation examples for the Bollini et al. (2013) dataset. The scrape action is required for this to be a valid plan, otherwise the dough would not be transferred to the baking sheet and oven.

## 3.4 Discussion

Evaluations showed that our method improved plan validity as measured by the mean number of precondition errors, syntax errors, and accuracy of steps returned (F1) in each plan. LCS remained fairly constant across our evaluations and ablations. The LCS metric reflects both the content of planning steps and their sequencing. By contrast, F1 only assesses the accuracy of steps in the generated plans. There may exist a trade-off wherein the process of inserting corrective plan steps reduces the amount of alignment of the generated and ground truth plans (lowering LCS), but increases the accuracy of included steps (raising F1). Of all the metrics considered, our method resulted in the largest reduction in the number of precondition errors (PE). We achieved these improvements without significant reductions in LCS and with an increase in F1. This is an important validation for our method,

as Huang et al. (2022) found that there exists a trade-off between the executability and semantic correctness (measured by LCS) of generated plans. It is straightforward to increase executability (fewer precondition errors) by ignoring the instructional text content and only outputting valid actions. For any downstream applications, plans must be executable while also reflecting the content of the instructions. Therefore, it is important to reduce the number of precondition errors while maintaining content similarity to ground-truth plans.

## 3.5 Conclusions

We develop an approach to semantic parsing for long-form instructional texts that leverages planning domain information to generate more valid plans in a low-data, few-shot setting. Our method significantly reduces the number of precondition errors present in semantically parsed plans for two recipe datasets. These results highlight the benefit of a neuro-symbolic approach that utilizes the code-generation LLM Codex to produce relevant steps for recipe execution and refines these plans using classical symbolic planning. In quantitative and qualitative evaluations, our approach generates plans that reflect the relevant steps of the natural language recipe. The symbolic planning component corrects precondition errors that arise from omitted or implied instructional steps and the challenges of learning with long context-dependencies from limited examples.

## Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001122C0007. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

# Chapter 4: Compositional Instruction Following with Language Models and Reinforcement Learning

*Presented at the Reinforcement Learning Conference (RLC) 2025. Published in Transactions on Machine Learning Research (TMLR) 2024.*

## 4.1 Introduction

Action reasoning in sequential decision-making often requires learning from interaction rather than from a hand-specified domain. Chapter 3 addresses executability by leveraging an explicit planning domain, but such formal specifications are unavailable in many settings. Without them, an agent must discover how actions change environment states through trial and error, and do so efficiently enough to scale to the combinatorially many tasks expressible in natural language. This chapter tackles that sample-efficiency problem by exploiting compositionality: because language and behavior share compositional structure, task policies can be represented as logical compositions of reusable value-function primitives rather than learned independently for each instruction. The result is a second method for improving action reasoning, complementary to the planner-based approach in Chapter 3, that enables zero-shot generalization to novel instructions.

An important goal of reinforcement learning (RL) is the creation of agents capable of generalizing to novel tasks. Natural language provides an intuitive way to specify a variety of tasks, and natural language has important properties: its structure is compositional and often mirrors the compositional structure of the tasks being described. To address the challenge of combining reinforcement learning with language grounding, in which the agent needs to explore the environment while simultaneously learning multiple language-conditioned

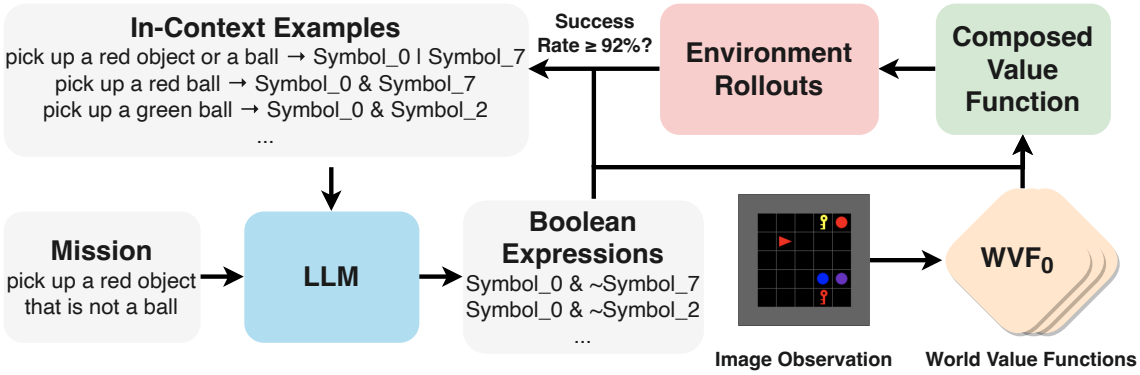



Figure 4.1: Pipeline diagram of the learning process for the CERLLA agent. The agent takes in a BabyAI language mission command and a set of 10 in-context examples that are selected using the BM25 search retrieval algorithm (Robertson and Zaragoza, 2009). The agent produces 10 candidate Boolean expressions. These expressions specify the composition of the base compositional value functions. Each compositional value function is instantiated in the environment and the policy it defines is evaluated over 100 rollouts. If the success rate in reaching the goal is greater than 92%, the expression is considered a valid parse of the language instruction and is added to the set of in-context examples.

tasks, we introduce a novel method: the compositionally-enabled reinforcement learning language agent (CERLLA). Our method reduces the sample complexity of tasks specified with language by leveraging compositional policy representations and a semantic parser trained using reinforcement learning and in-context learning.

While previous works have attempted to use natural language to specify tasks for RL agents (Ahn et al., 2023; Blukis et al., 2020), here we exploit the compositional nature of language along with compositional policy representations to demonstrate improvements in sample complexity and generalization in solving novel tasks.

Previous approaches to mapping language to behaviors use policies learned using imitation learning (Silva et al., 2021; Ahn et al., 2023; Blukis et al., 2020). In this work, we instead focus on the setting where the agent does not have access to supervised demonstrations and instead learns to ground language to specified behaviors with RL. The challenge in this approach is the significantly higher sample complexity of RL-based methods when

grounding behaviors. Agents must map a variety of potential language instructions to unknown corresponding behaviors. Pretraining and transfer learning offers one possible solution. In natural language processing, pretraining language models such as BERT (Devlin et al., 2019) and GPT-4 (OpenAI, 2023) have enabled substantial reductions in sample complexity of solving novel NLP tasks. However, in RL there is a lack of pretrained policy representations that can be fine-tuned using novel examples in few-shot settings.

In CERLLA, "pretraining" instead involves learning representations that can be composed to solve novel tasks (Todorov, 2009; Nangue Tasse et al., 2020). For instance, Nangue Tasse et al. (2020) demonstrate zero-shot task solving using compositional value functions and Boolean task algebra. The value functions are composed using Boolean operators to produce new complex behaviors. But these methods require manual specification of the Boolean expressions that describe value function composition, thus limiting their application to novel tasks. We overcome this and use RL to learn to compose the value functions, given a task description in natural language.

Our method uses these compositional value functions and pretrained language models to solve a large number of tasks using RL while not relying on curricula, demonstrations, or other external aids to solve novel tasks. Leveraging compositionality is essential to solving large numbers of tasks with shared structure. The sample complexity of learning a large number of tasks using RL is often prohibitive unless methods leverage compositional structure (Mendez-Mendez and Eaton, 2023).

This work builds on the Boolean compositional value function representations of Nangue Tasse et al. (2020) to construct a system for learning compositional policies for following language instructions. Our insight is that language commands reflect the compositional structure of the environment, but without compositional RL representations, this structure cannot be used effectively. Language, therefore, unlocks the utility of

compositional RL, allowing us to not only compose base policies but also negate specific behaviors to solve tasks such as "Don't start the oven." These language-conditioned compositional RL policies can be used as pretrained general-purpose policies, and novel behaviors can be added as needed when solving new tasks. Moreover, the composed policies themselves are interpretable as we can inspect the base policies from which they are composed.

## 4.2 Domain

Because we build on the compositional value function representations of Nangue Tasse et al. (2020), our method is applicable to any environment with goal-reaching tasks, the ability to learn value functions through RL, and language instructions. To evaluate our method, we select the BabyAI MiniGrid domain (Chevalier-Boisvert et al., 2019), an easily extensible test-bed for compositional language-RL tasks used in many recent language-RL works including (Carta et al., 2022; Li et al., 2022). It has language commands, image-state observations, a discrete action space, and objects with both color and type attributes. We augment BabyAI with $162$ compositional tasks specified using intersection, disjunction, and negation. Table 4.1 provides a full list of task attributes available in the environment and the grammar of the Boolean expressions. Table 4.2 provides examples of the types of tasks our method learns. The environment is initialized with one or more goal objects and distractor objects that are randomly placed.

For each episode in the BabyAI environment, the agent is provided with two forms of input as observations: a task instruction formulated in natural language and a $56 \times 56 \times 3$ RGB image representing the state of the environment at each time-step. The objective for the agent is to correctly identify and pick up a specific goal object, which is described by the task instruction. The goal objects are defined by their attributes: three shapes and six

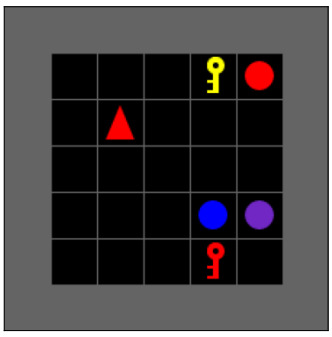


Figure 4.2: Example of a task in the BabyAI domain (Chevalier-Boisvert et al., 2019). The agent (red triangle) needs to complete the mission – "pick up the red key". Solving this task with compositional value functions requires using the conjunction of the *pickup* "red object" and "key" value functions.

colors. The combination of these attributes results in 18 unique goal objects that the agent can pick up. In each episode, the environment contains at least one correct goal object and four additional "distractor" objects, which are sampled randomly.

We consider the case of an agent required to solve a series of related tasks. Each task is formalized as a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the set of actions available to the agent. The transition dynamics $p(s'|s,a)$ specify the probability of the agent entering state $s'$ after executing action $a$ in state $s$, while $r(s,a,s')$ is the reward for executing $a$ in $s$. We further assume that $r$ is bounded by $[r_{\text{MIN}}, r_{\text{MAX}}]$. We focus here on goal-reaching tasks, where an agent is required to reach a set of terminal goal states $\mathcal{G} \subseteq \mathcal{S}$.

Tasks are related in that they differ only in their reward functions. Specifically, we first define a background MDP $M_0 = \langle \mathcal{S}_0, \mathcal{A}_0, p_0, r_0 \rangle$. Then, any new task $\tau$ is characterized by a task-specific reward function $r_\tau$ that is non-zero only for transitions entering $g$ in $\mathcal{G}$. Consequently, the reward function for the resulting MDP is given by $r_0 + r_\tau$.

We implement this reward function with a penalty of $r_0 = -0.1$ for each step taken.

If the agent chooses the `pickup` action upon reaching an object, it observes the picked object, and the episode ends. If the selected object matches the correct goal as per the task instruction, the agent receives a reward of $r_\tau = +2$.

The agent aims to learn an optimal policy $\pi$, which specifies the probability of executing an action in a given state. The value function of policy $\pi$ is given by $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} r(s_t, a_t)\right]$ and represents the expected return after executing $\pi$ from $s$. Given this, the optimal policy $\pi^*$ is that which obtains the greatest expected return at each state: $V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s)$ for all $s \in \mathcal{S}$. Closely related is the action-value function, $Q^\pi(s, a)$, which represents the expected return obtained by executing $a$ from $s$, and thereafter following $\pi$. Similarly, the optimal action-value function is given by $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.

## 4.3 Approach

We propose a two-step process for training an RL agent to solve Boolean compositional tasks with language. During an initial pretraining phase, a set of WVFs are learned which can later be composed to solve new tasks in the environment. This set forms a task basis that can express any task which can be written as a Boolean algebraic expression using the WVFs.

In a second phase, an LLM learns to semantically parse language instructions into the Boolean compositions of WVFs using RL and in-context learning. The parser learns this mapping from abstract symbols to WVFs using RL by observing language instructions and interacting with the environment. Notably, our method does not require the semantic parser to have access to any knowledge of the underlying basis tasks that the WVFs represent, and instead regards the WVFs as abstract symbols which can be composed to solve tasks. Since the semantic parser does not have access to any information about what task a WVF

represents, our method can be applied in principle to any basis of tasks.

Tasks like "pickup the red key" can be represented by taking the intersection of the WVFs for picking up "$red$" objects and "$key$" objects: $red \wedge key$. Our method also supports negation and disjunction: we can specify tasks like "pick up a red object that is not a ball": $red \wedge \neg ball$. We augment this domain with additional tasks. For further examples of tasks, see Table 4.2, which lists the complete set of tasks created using the attributes $yellow$ and $key$. We implement the model from Chevalier-Boisvert et al. (2019) as a non-compositional baseline. This model does not have a pretraining phase for its RL representations, and in our experiments we account for this discrepancy in training steps by penalizing our agent by the number of training steps needed to learn the WVFs.

### 4.3.1 World Value Functions

#### 4.3.1.1 Logical Composition of Tasks using World Value Functions (WVFs)

Recent work (Nangue Tasse et al., 2020, 2022) has demonstrated how logical operators such as conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$) can be applied to value functions to solve semantically meaningful tasks compositionally with no further learning. To achieve this, the reward function is extended to penalize the agent for attaining goals it did not intend to:

$$\bar{r}(s,g,a) = \begin{cases} \bar{r}_{MIN} & \text{if } g \neq s \in \mathcal{G} \\ r(s,a) & \text{otherwise,} \end{cases} \tag{4.1}$$

where $\bar{r}_{MIN}$ is a large negative penalty. The agent receives the unmodified reward $r(s,a)$ for all steps except where it reaches a different goal state than intended: $g \neq s \in \mathcal{G}$. Given $\bar{r}$, the related value function, termed *world value function (WVF)*, can be written as: $\bar{Q}(s,g,a) = \bar{r}(s,g,a) + \int_{\mathcal{S}} \bar{V}^{\bar{\pi}}(s',g)p(s'|s,a)ds'$, where $\bar{V}^{\bar{\pi}}(s,g) = \mathbb{E}_{\bar{\pi}}\left[\sum_{t=0}^{\infty} \bar{r}(s_t,g,a_t)\right]$.

These value functions are intrinsically *compositional* since if a task can be written as a logical expression over previous tasks, then the optimal value function can be similarly derived

by composing the learned WVFs. For example, consider the `PickUpObject` environment shown in Figure 4.1. Assume the agent has separately learned the task of collecting red objects (task $R$) and keys (task $K$). Using these value functions, the agent can immediately solve the tasks defined by their union ($R \vee K$), intersection ($R \wedge K$), and negation ($\neg R$) as follows: $\bar{Q}^*_{R \vee K} = \bar{Q}^*_R \vee \bar{Q}^*_K \coloneqq \max\{\bar{Q}^*_R, \bar{Q}^*_K\}$, $\bar{Q}^*_{R \wedge K} = \bar{Q}^*_R \wedge \bar{Q}^*_K \coloneqq \min\{\bar{Q}^*_R, \bar{Q}^*_K\}$, and $\bar{Q}^*_{\neg R} = \neg \bar{Q}^*_R \coloneqq \left(\bar{Q}^*_{MAX} + \bar{Q}^*_{MIN}\right) - \bar{Q}^*_R$, where $\bar{Q}^*_{MAX}$ and $\bar{Q}^*_{MIN}$ are the world value functions for the *maximum* and *minimum* tasks respectively.[1]

#### 4.3.1.2 Pretraining World Value Functions

During pretraining, a set of WVFs is learned which can later be composed to solve any task in the BabyAI environment. Each WVF takes as input a $56 \times 56 \times 3$ RGB image observation of the environment and outputs $|\mathcal{G}| \times |\mathcal{A}| = 18 \times 7$ values for accomplishing one of the basis tasks (by maximizing over the goal-action values). As there are nine object attributes (three object type attributes and six color attributes as listed in Table 4.1) we train nine WVFs. Each WVF is a value function for the policy of picking up objects that match one of the nine attributes. However, our method does not require knowledge of the underlying semantics of the value functions (i.e. the names of the underlying task attributes). We therefore assign each WVF a random identifier, denoted as $Symbol\_0$ through $Symbol\_8$. While we refer to the WVFs by their color and object type in the paper, our model does not have access to this information and only represents the WVFs by their unique identifiers.

Each WVF is implemented using $|\mathcal{G}| = 18$ CNN-DQN (Mnih et al., 2015) architectures. The WVF pretraining takes nineteen million steps. This is done by first training $\bar{Q}^*_{MIN}(s, g, a)$ for one million steps and $\bar{Q}^*_{MAX}(s, g, a)$ for eighteen million steps (one million

[1]The maximum task is defined by the reward function $r = r_{\text{MAX}}$ for all $\mathcal{G}$. Similarly, the minimum task has reward function $r = r_{\text{MIN}}$ for all $\mathcal{G}$.

steps per goal in the environment). Each basis WVF $\bar{Q}^*_B(s, g, a)$ is then generated from $\bar{Q}^*_{MIN}(s, g, a)$ and $\bar{Q}^*_{MAX}(s, g, a)$ by computing $\bar{Q}^*_B(s, g, a) = \bar{Q}^*_{MAX}(s, g, a)$ **if** $r_B(g, a) = r_{\text{MAX}}$ **else** $\bar{Q}^*_{MIN}(s, g, a)$. This yielded a $98\%$ success rate for each basis WVF. For more details on WVF pretraining see Section 4.3.1.1 and Nangue Tasse et al. (2022), and see Tables 4.3 and 4.4 for a full list of hyperparameters used in WVF training.

Table 4.1: Task attributes and Boolean grammar. The symbols uniquely identify each learned WVF.

| **Task Attributes** | | **Boolean Grammar** | |
|---|---|---|---|
| **Colors** | **Objects** | **Symbols** | **Operators** |
| $red$, $purple$, $grey$ $green$, $yellow$, $blue$ | $key$, $ball$, $box$ | $Symbol\_0$, $Symbol\_1$,. . . ,$Symbol\_8$ | AND: $\&$ OR: $\mid$, NOT: $\sim$ |

Table 4.2: Example language instructions and corresponding Boolean expressions for the $yellow$ and $box$ attributes.

| **Language Instruction** | **Ground Truth Boolean Expression** |
|---|---|
| pick up a yellow box | $yellow \,\&\, box$ |
| pick up a box that is not yellow | $\sim yellow \,\&\, box$ |
| pick up a yellow object that is not a box | $yellow \,\&\, \sim box$ |
| pick up an object that is not yellow and not a box | $\sim yellow \,\&\, \sim box$ |
| pick up a box or a yellow object | $yellow \mid box$ |
| pick up a box or an object that is not yellow | $\sim yellow \mid box$ |
| pick up a yellow object or not a box | $yellow \mid \sim box$ |
| pick up an object that is not yellow or not a box | $\sim yellow \mid \sim box$ |
| pick up a box | $box$ |
| pick up an object that is not a box | $\sim box$ |
| pick up a yellow object | $yellow$ |
| pick up an object that is not yellow | $\sim yellow$ |

### 4.3.2 Compositionally Enabled RL Language Agent (CERLLA)

We assume the downstream tasks are distinct from the basis tasks. During downstream task learning, the pretrained WVFs are composed to solve novel tasks specified in language.

Table 4.3: Hyperparameters for the LLM Agent.

| LLM Agent Hyperparameters | |
|---|---|
| LLM | GPT-4 and GPT-3.5 |
| Beam Width | 10 |
| Rollouts | 100 episodes |
| In-Context Examples | 10 |
| Training Temperature | 1.0 |
| Evaluation Temperature | 0.0 |

Table 4.4: Hyperparameters for world value function pretraining. The Adam optimizer was introduced by Kingma and Ba (2015).

| WVF Learning Hyperparameters | |
|---|---|
| Optimizer | Adam |
| Learning rate | 1e-4 |
| Batch Size | 32 |
| Replay Buffer Size | 1e3 |
| $\epsilon$ init | 0.5 |
| $\epsilon$ final | 0.1 |
| $\epsilon$ decay steps | 1e6 |

To solve BabyAI language-specified tasks, the agent must interpret the input language command and pick up an object of an allowed type. To accomplish this, the semantic parser maps from language to a Boolean expression specifying the composition of WVFs. These Boolean expressions are then composed using a fixed pipeline that takes as input the set of WVFs and the Boolean expression. This pipeline parses the Boolean expression and returns a composed WVF. The agent then acts in the environment under the policy of the WVF by taking the action with the greatest value at each step. If the Boolean expression is not syntactically correct, it cannot be instantiated as a WVF and the episode terminates unsuccessfully.

Table 4.5: The prompting strategy for the CERLLA semantic parsing module.

| Role | Content |
|---|---|
| *System* | "We are going to map sentences to Boolean expressions. The Boolean expression variable Symbols are numbered 0 to 8, e.g. $Symbol\_0$, $Symbol\_1$... The operators are and : &, or : \|, not : ~. I will now give a new sentence and you will come up with an expression. Now wait for a new sentence command. Respond with a list of 10 candidate Boolean expressions. Respond only with the list of Boolean expressions. Never say anything else." |
| *User (Example)*<br>*Assistant* | "pick up a red ball"<br>"$Symbol\_0$ &$Symbol\_7$" |
| | [Additional in-context examples] |
| *User (Command)*<br>*Assistant* | "pick up a red object that is not a ball"<br>"$Symbol\_0$ & $Symbol\_1$ & $\sim Symbol\_2$"<br>"$Symbol\_3$ & $\sim Symbol\_4$"<br>"$Symbol\_5$ & $Symbol\_6$ & $\sim Symbol\_7$"<br>[Additional candidate expressions] |

#### 4.3.2.1 In-Context Semantic Parsing with RL

To implement the semantic parser, we utilize the large language models GPT-4 (OpenAI, 2023) and GPT-3.5.[2] Our method builds on the work of Shin et al. (2021), which builds a semantic parser using LLMs and few-shot learning, and Toolformer (Schick et al., 2023), which learns an LLM semantic parser from weak supervision. Our method is distinct from these approaches in that it utilizes in-context examples combined with an environment rollout RL signal. At the start of learning, the agent has no in-context examples of valid mappings from language to Boolean expressions (see Figure 4.1). At each episode, our agent is prompted with general instructions defining the semantic parsing task, the input language command, and up to $10$ previously-acquired in-context examples selected using the BM25 retrieval algorithm (Robertson and Zaragoza, 2009).

[2]https://platform.openai.com/docs/models/gpt-3-5

During training, the LLM is sampled with a temperature of $1.0$ and produces a beam of $10$ semantic parses (Boolean expressions) of the input language command. Together the temperature and beam width control the exploitation-exploration trade-off of the semantic parsing model. Each candidate Boolean expression is parsed using a fixed pipeline and instantiated as a WVF. The policy defined by the WVF is evaluated in the environment over $100$ episode rollouts. If the success rate across these episodes in reaching the specified goals is greater than or equal to $92\%$, the language instruction and Boolean expression are added to the list of in-context examples. Multiple Boolean expressions may attain high reward for any given task. To counter this we add a form of length-based regularization. If the agent already has an in-context example with the same language instruction, the length of the Boolean expressions is compared and only the shorter of the two expressions is retained as an in-context example. We thereby favor shorter Boolean expressions that attain high reward in the environment. For more details of the prompting strategy, see Table 4.5.

### 4.3.3 Baselines

The baseline is a joint language and vision model which learns a single action-value function for all tasks based on the architecture used in the original BabyAI paper (Chevalier-Boisvert et al., 2019). We explore two baseline models: an LM Baseline that utilizes pretrained language representations for embedding mission commands from a frozen "all-mpnet-base-v2" model from the SentenceTransformers library (Reimers and Gurevych, 2019) based on the MPNet model (Song et al., 2020) and an ablated Baseline which does not use pretrained language representations. This pretrained sentence embedding model is trained on diverse corpora of sentence embedding tasks.

## 4.4 Experiments

We conduct experiments across four agent types and two settings. The first experiment evaluates sample complexity (Figure 4.3). We learn all 162 tasks simultaneously and plot the mean success rate against the number of environment steps. The second experiment divides the task set in half, and measures the ability of the agents to generalize to held-out novel tasks while learning from a fixed set of tasks (Figure 4.5).

We evaluate our method implemented using both GPT-4 and GPT-3.5. We compare our method to the baseline agents, but penalize our method by the number of environment steps required to learn the pretrained WVFs. As an upper limit on the performance of our method, we compare to an Oracle Agent which has a perfect semantic parsing module. It has access to the ground-truth mappings from language to Boolean expressions and its performance is limited only by the accuracy of the pretrained policies and randomness in the environment.

### 4.4.1 Simultaneous Learning of 162 Tasks

In this experiment, at each episode a random task is sampled from the set of 162 language tasks. The baseline agents learn for 21 million steps, and CERLLA learns for 2 million steps. Because our agent pretrains the WVFs, we penalize our agent by starting it at 19 million steps (Figure 4.3). Note that this disadvantages our method, as the WVF pretraining phase does not include language information and its only exposure to language-task data is over the following two million steps. Our method therefore has access to less information about the task structure than the baseline agents during the first 19 million steps. For CERLLA, due to the latency and cost of invoking the LLM, we only evaluate on one randomly selected task every 5,000 environment steps, computing the average performance over 100 episodes. For the baseline agents we evaluate all 162 tasks every 50,000 timesteps.

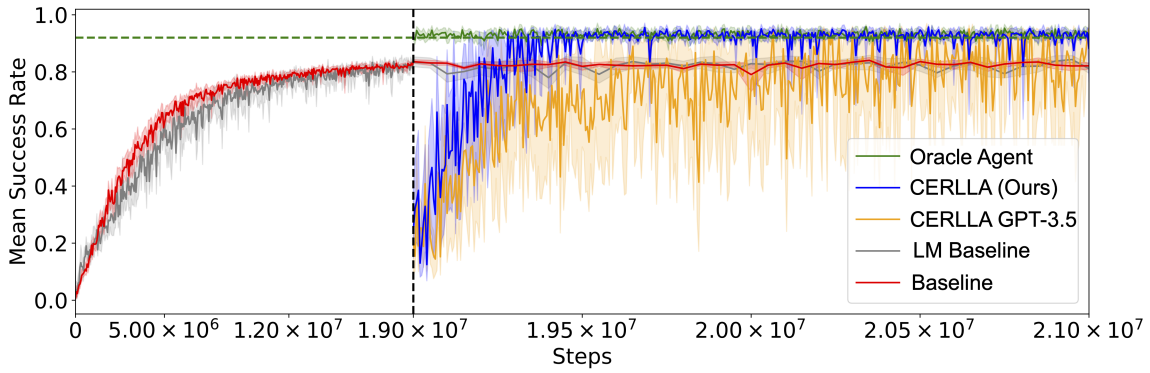



Figure 4.3: Results for learning all 162 tasks simultaneously. The mean episode success rate is plotted against the number of environment steps. Learning curves are presented for CERLLA and the non-compositional baseline agents. The Oracle agent is given the ground-truth Boolean expressions and upper bounds the attainable success rate in the environment, denoted by the dashed line at 92%. Our method is initialized at 19 million steps to reflect the number of training steps used in pretraining the compositional value functions. Note the change in steps scale at 19 million steps. Means and 95% confidence intervals are reported over 10 trials, 5 trials for the LM Baseline.

This results in higher variance for our method in the plots.

We also plot the number of in-context training examples added to CERLLA's set in Figure 4.4. This is equivalent to the number of training tasks successfully solved at that step. The Oracle Agent solves the overwhelming majority of tasks during their first occurrence and is limited only by the small amount of noise in the policies and environment.

### 4.4.2 Held-out Task Generalization

This experiment (Figure 4.5) measures the generalization performance of each method on held-out tasks. We compare the performance of CERLLA to the baseline agents. In this setting, the set of tasks is randomly split into two halves at the start of training. At each episode, a random task from the first set is selected. During evaluation of our agent, one random task from each set is selected and the agent is evaluated over 100 episodes. The baseline agents are evaluated over all 81 tasks in each set.

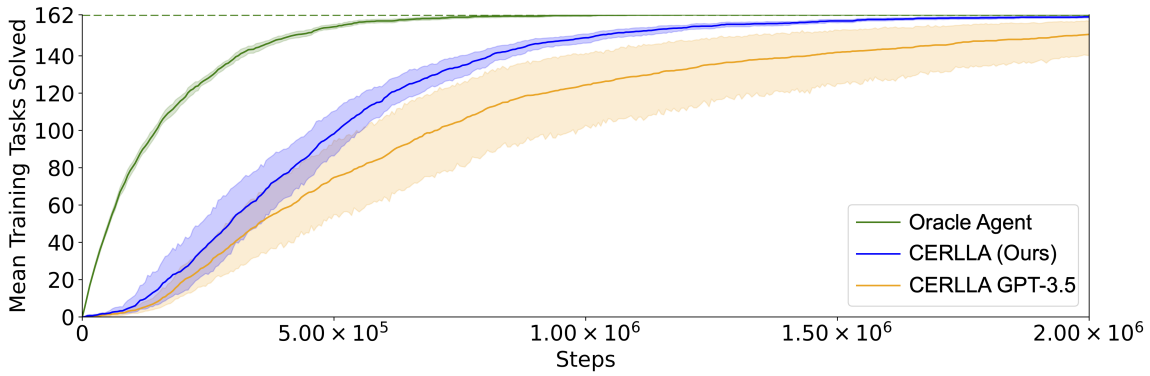



Figure 4.4: The mean number of tasks solved is plotted against the number of environment steps. This quantity is equal to the total number of in-context examples present in CERLLA's in-context example set at that step. Because the Oracle Agent has access to the ground truth Boolean expressions for each task, it solves tasks immediately. The population of tasks remains constant, so the number of unsolved tasks decreases over time, leading to a logistic learning curve and an exponential decay in the rate at which new tasks are solved for the Oracle Agent. Means and $95\%$ confidence intervals are reported over $10$ trials.

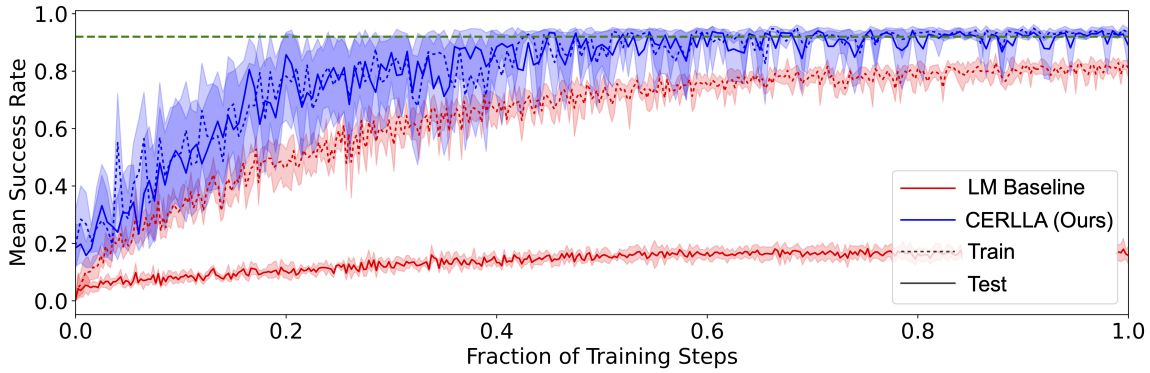



Figure 4.5: Generalization performance for learning on $81$ randomly selected tasks (train) and evaluating on the held-out $81$ tasks (test). Learning curves are presented for our method and the baseline agents. The dashed and solid lines represent performance on the training and test sets respectively. The $x$-axis represents the fraction of the total training steps completed: $1$ million for the CERLLA and $21$ million for the LM Baseline agent. The dashed line denotes the success rate for considering the environment solved at $92\%$. Means and $95\%$ confidence intervals are reported over $15$ trials, $5$ trials for the LM Baseline.

## 4.5 Discussion

In both experiments CERLLA attains a significantly higher success rate in fewer total samples than the baselines. Figure 4.3 shows our method attains a $92\%$ success rate (matching the performance upper bound of the Oracle Agent) after only $600k$ environment steps, a small fraction of the steps of the baseline. The baseline agents are not able to generalize to all $162$ tasks and only reach a success rate of $\approx 80\%$ after $21$ million steps. Note that while the WVF pretraining for our method requires $19$ million steps, the pretraining objective does not include any language instructions and is distinct from the downstream task objective. The baseline learns the downstream tasks and language for its first $19$ million training steps and still does not solve the $162$ tasks. Even though the baseline agent is given access to the language commands and reward structure for the entire duration of its training, the compositional agent is still able to outperform the baseline. These results show the necessity of compositional representations for being able to learn large numbers of compositional tasks in a sample efficient manner.

Figure 4.5 demonstrates that our method is able to generalize well to held-out tasks. The performance of the agent on training tasks and held-out tasks is very similar. This is expected given the ability to generalize compositionally in both the policy and language spaces. The LM Baseline agent cannot generalize well between the training and held-out tasks as it lacks the necessary compositional representations. Note that this experiment trains the CERLLA and LM Baseline agent for fewer steps than the $162$ task experiment: one million and $21$ million steps respectively. There is a significant generalization gap for this agent even after $21$ million steps, significantly more steps than our method, even accounting for WVF pretraining steps.

In the $162$ task learning experiment, CERLLA using GPT-3.5 does not exceed the performance of the baselines even after two million steps indicating poor compositional

generalization. Confirming this, Figure 4.4 shows the mean number of tasks solved, which is the same as the total number of potential in-context examples that can be selected from during inference. Despite a weaker version of CERLLA solving most of the tasks, this does not transfer to a high evaluation success rate in Figure 4.3. The variance of the CERLLA agent using GPT-3.5 is higher than with GPT-4. This is caused by relatively worse generalization from available in-context examples than GPT-4. This indicates that more powerful language models are likely to improve the performance of CERLLA, up to the limit of the oracle agent. Highlighting the interpretability of our language-RL learning framework, we provide a qualitative analysis of the cause of the GPT-3.5 agent's lower performance in Table 4.6.

| **Language Instruction** | **Boolean Expression** |
|---|---|
| pick up a ball or a grey object | $Symbol\_0 \mid Symbol\_4$ |
| pick up a box that is not grey | $\sim Symbol\_4 \,\&\, Symbol\_2$ |
| pick up a grey ball | $Symbol\_0 \,\&\, Symbol\_4$ |
| pick up a ball or an object that is not grey | $(Symbol\_0 \mid Symbol\_1) \mid \sim Symbol\_4$ |
| pick up a ball that is not grey | $Symbol\_0 \,\&\, \sim Symbol\_4$ |
| pick up a grey object or not a ball | $(Symbol\_0 \,\&\, Symbol\_4) \mid \sim Symbol\_0$ |
| pick up a grey object that is not a ball | $\sim Symbol\_0 \,\&\, Symbol\_4$ |
| pick up a grey object or not a key | $(Symbol\_0 \,\&\, Symbol\_1) \mid$ $(\sim Symbol\_2 \,\&\, Symbol\_4) \mid \sim Symbol\_5$ |
| pick up an object that is not grey | $(Symbol\_0 \,\&\, Symbol\_1) \mid \sim Symbol\_4$ |
| pick up an object that is not grey and not a ball | $\sim Symbol\_4 \,\&\, \sim Symbol\_0$ |

Table 4.6: Subset of the in-context examples that the GPT-3.5 agent has accumulated at the end of 2 million steps of learning the 162 tasks. As shown, many of these expressions are not consistent or needlessly complicated. This helps to explain the relatively poorer performance of the GPT-3.5 agent which produces much noisier semantic parses and thus has much higher variance and lower performance than the GPT-4 agent. Reducing the sampling temperature during learning leads to better expressions, but at the cost of slower exploration and learning.

Overall, our method solves a challenging set of 162 language-RL tasks, which to our knowledge is the largest set of simultaneously-learned language-RL tasks. In the real world, many tasks share compositional structure, and language has inherent compositional structure. Therefore, it is important to develop methods which leverage this shared compositionality to

reduce the sample complexity of acquiring new behaviors. This is critical for reinforcement learning problems that already have high sample complexity due to the inefficiency of learning from RL feedback. We address these challenges using a novel semantic parsing method, which uses in-context and environment feedback to learn to map natural language instructions to Boolean expressions specifying the composition of value functions. Our method significantly outperforms a non-compositional baseline approach, demonstrating the importance of methods that can generalize compositionally in terms of both language and RL representations.

## 4.6 Conclusions

We introduce a method that integrates pretraining of compositional value functions with large language models to solve language tasks using RL. Our method rapidly solves a large space of RL tasks specified in language more effectively than previous approaches. Demonstrating efficacy across 162 tasks with reduced sample requirements, our findings also further differentiate the capabilities of more powerful language models from weaker ones in the task of semantic parsing using an environment policy rollout signal. The combination of compositional RL with language models provides a novel framework for reducing the sample complexity of learning RL tasks specified in language.

## 4.7 Author Contributions

This work was done jointly with co-first author Geraud Nangue Tasse while he was a PhD student at The University of the Witwatersrand in South Africa. The CERLLA method builds on the World Value Functions introduced by Geraud in Nangue Tasse et al., 2020, 2022. Geraud trained the World Value Functions used in this work and advised on experiment design and baseline implementations. Vanya Cohen devised the CERLLA method for combining these pretrained value functions to solve tasks specified in language using few-shot semantic parsing learned from environment feedback.

## Acknowledgments

This material is based upon work supported by DARPA's Perceptually-enabled Task Guidance (PTG) program under Contract No. HR001122C0007. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. This material is also based upon work supported by the Air Force Office of Scientific Research under award number FA9550-24-1-0239. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force. This work was also supported by ONR grant #N00014-23-1-2887 and #N00014-19-2076.

# Chapter 5: CaT-Bench: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans

*Presented at the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2024.*

## 5.1 Introduction

Action reasoning requires understanding how actions depend on one another: which preconditions must hold, which effects afford later steps, and which orderings are therefore valid. In Chapter 3, we use external information in the form of planning domain definitions to improve action precondition satisfaction and generate more valid plans from instructional texts. However, constructing such formal domain specifications is costly and domain-specific. As foundation models increasingly serve as the core representations for decision-making agents (Cohen et al., 2024), for many real-world applications it is more practical to rely on the language model's own internalized knowledge of action preconditions, effects, and valid orderings rather than requiring a hand-crafted planning domain for every new setting. This motivates the need to evaluate the degree to which frontier language models possess this understanding. In this chapter, we assess the ability of frontier language models to identify causal and temporal dependencies between the steps of natural-language plans without access to external information. In Lal et al. (2024), we construct CaT-Bench for this purpose. We modify the Recipe Flow Graph Corpus (Yamakata et al., 2020), containing recipes with sub-step procedure dependencies, to produce a question-based benchmark of 4260 dependency questions.

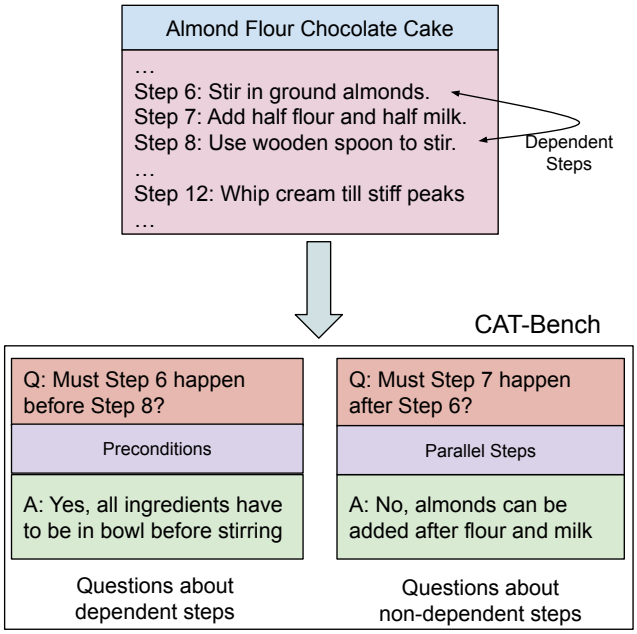



Figure 5.1: We use step-pair dependency annotations to create CaT-Bench, a question-driven evaluation framework for plan-based reasoning. Questions in this benchmark elicit reasoning about different causal relations such as preconditions, effects and step independence.

Planning is central to decision making and has been studied in various domains such as robotics and embodied environments (LaValle, 2006; Jiang et al., 2019). To follow, revise, or customize a plan, one must be able to reason about the steps involved as well as their causes and effects (Brahman et al., 2023). Recent work on evaluating reasoning in plans focuses on classical problems such as Blocksworld (Slaney and Thiébaux, 2001; Valmeekam et al., 2023), simulated environments like AlfWorld (Shridhar et al., 2021b), or restricted language such as PDDL (Zhang et al., 2024). However, real-world natural language plans cannot be executed to test for correctness and reliability. This chapter describes a question-driven evaluation to better study the detailed causal and temporal connections within such plans.

Given a plan, such as making a cake in Figure 5.1, one must understand its various aspects to answer questions about it. Answering if ground almonds should be added before

stirring the mixture requires understanding that a *precondition* for mixing evenly is that all ingredients should be added already. But reasoning about whether flour should be added after almonds requires figuring out *step independence* since the order of adding ingredients does not matter here. Such causal aspects are encoded in temporal step dependencies of plans.

West et al. (2024) show that LLMs create expert-like outputs, but their generative capability does not necessarily indicate a correspondingly strong capability to understand underlying phenomena. While LLMs appear to generate good plans, it is unclear how well they understand important aspects of the steps themselves. We thus use CaT-Bench to test whether LLMs can identify step dependencies that reflect the causal and temporal structure of the plan. We find that current LLMs struggle to identify step dependencies, often performing close to random chance, raising more questions about their understanding of instructional text.

Using notions of consistency to evaluate their robustness, we also show that almost all out-of-the-box LLMs are largely unreliable. Few-shot prompting with retrieved exemplars improves performance and consistency ($0.49 \rightarrow 0.68$ F1 for `gpt-4o`). Explanation-based generation offers another route to improve model performance and reliability on reasoning tasks (Camburu et al., 2018). Prompting LLMs to explain their decisions also improves performance on CaT-Bench ($0.49 \rightarrow 0.7$ F1). Despite these gains, there is still large room for improvement in identifying step dependencies. When also considering the quality of the explanations, the average human rating for satisfactory answers from state-of-the-art LLMs is only $\sim$3 (out of 5). Further, contrary to prior findings, using chain-of-thought prompting (CoT), i.e., reasoning before answering (Wei et al., 2022b), performs worse than answering first and then explaining it, indicating inconsistencies in model reasoning.[1]

[1]CaT-Bench is available at `https://huggingface.co/datasets/vanyacohen/CaT-Bench` and the

## 5.2 CaT-Bench

Understanding plans requires reasoning about how different actions in a plan relate to each other. In this work, we focus on the ability to recognize *temporal dependencies* between steps i.e., deciding if a step *must* happen before another. Typically, step $i$ must happen before a step $j$ if the effects (outcomes) of step $i$ satisfy one or more preconditions necessary for the proper execution of step $j$, or if the effects of step $j$ aggregate or modify the effects of step $i$ in service of accomplishing a (sub-)goal. Examples of the types of tasks in CaT-Bench are shown in Figure 5.1. Thus, recognizing such dependencies requires the ability to infer many important logical connections such as preconditions, causes, sub-goals, and effects of the steps. This suggests that a simple test of whether a step must happen before another step (or after) can be an effective test of reasoning about the various logical dependencies between the steps in a plan.

We build on this idea to create CaT-Bench, a new dataset of causal dependency questions defined over cooking recipes. Specifically, we make use of the Recipe Flow Graph Corpus (Yamakata et al., 2020) containing 300 English language cooking recipes annotated with substep procedure dependencies. For each recipe, this dataset provides a directed acyclic graph (DAG), in which the nodes are steps and directed edges indicate the temporal edge between those steps. If the nodes corresponding to two steps are not connected by a directed path, then they can be performed in any order (with respect to themselves) without changing the recipe result. In other words, two steps are temporally dependent if and only if there is a directed path from one to the other, and independent otherwise.

For all ordered pairs of steps $(i, j)$ in a plan, we create two binary yes/no questions: (i) ***Must*** $step_i$ ***happen before*** $step_j$*?* (ii) ***Must*** $step_j$ ***happen after*** $step_i$*?*. These questions

code is at `https://github.com/StonyBrookNLP/CaT-Bench`.

primarily test for precondition relations and the ability to understand effects of steps and how they relate to sub-goals or overall goals of the plan. We pool all such questions from dependent pairs of steps (i.e., the steps where there is a directed path from one step's node to the other in the recipe DAG) into DEP, and the rest into NONDEP.[2]

**Goal:** lemon zested strawberry shortcakes
**Steps:**
...
6. Divide dough in half.
7. Add sugar; beat until stiff peaks form.
8. Place 5cm apart on an ungreased baking tray.
9. Bake at 200 C / Gas 6 for 8-10 minutes.
10. Remove to a wire rack; cool for 15 minutes.
11. In bowl, combine butter and lemon zest; set aside.
12. In mixing bowl, beat cream until it begins to thicken.
13. Gently pat or roll each half into a 1.75cm thick circle.
14. To assemble, split shortcakes in half.
...
**DEP Q:** Must Step 8 happen before Step 10?
**DEP A:** Yes, removing a cake for cooling needs dough to be placed on a baking tray first.
**NONDEP Q:** Must Step 12 happen after Step 7?
**NONDEP A:** No, adding sugar is a part of making dough but beating the cream makes the filling.

Figure 5.2: Examples of different types of questions in a plan from CAT-BENCH. To correctly answer these questions, one must understand preconditions and effects (to answer DEP), some steps need not be performed in any particular order, and plans can contain subplans within them (to answer NONDEP).

In total, CAT-BENCH contains 2,840 questions about causal dependencies of steps for 57 unique plans. We undersample the non-dependent questions to ensure that NONDEP and DEP are of the same size (1,420 questions each). Half of CAT-BENCH tests the "before" temporal relation and the other half tests the "after" relation. It is therefore balanced in terms of both question types and temporal relation type. We also annotate the questions based

[2]The answers to all questions in the DEP set are 'yes', and the answers to NONDEP questions are 'no'.

on the distance between the pairs of steps. Two steps are deemed close if they are within 3 steps of each other, $(j - i) \leq 3$; there are 1,256 questions about *close* steps and 1,584 about *distant* steps.

CaT-Bench enables two tasks. Step Order Prediction elicits binary judgments about dependencies between pairs of steps in a plan, and performance on this task can be evaluated automatically. Step Order Explanation requires models to provide explanations for their judgments about step dependencies. This involves understanding causal relationships and expressing relevant knowledge about actions in the steps being asked about. Since this is a free-form generation task, these explanations require human evaluation. CaT-Bench does not contain gold, human-written explanations, and we advocate for reference-free human evaluation since there can be multiple valid explanations.

For Step Order Prediction, we evaluate model performance using standard precision, recall, and F1 score. We measure model robustness using temporal consistency: models must provide consistent predictions when asked before/after questions about the same step pair. If a model judges that $step_i$ happens before $step_j$, it must also judge that $step_j$ happens after $step_i$. We define this metric as Temporal Consistency (TC). For open-ended text generation in Step Order Explanation, we use human evaluation with a standardized interface to compare different models. We present answers from different models and ask crowd-workers on Amazon Mechanical Turk to assess their correctness on a 5-point Likert scale (Likert, 1932). For each plan, question, and model answer, we ask 3 distinct annotators to provide judgments. An explanation is considered invalid if it does not give a plausible reason that is also relevant to the question.

## 5.3 Experiments

We benchmark the performance of a variety of models on CaT-Bench. We evaluate `gpt-4-turbo`, `gpt-3.5-turbo`, `gpt-4o`, `claude-3.5-sonnet`, `gemini-1.0-pro`, `gemini-1.5-pro`, `gemini-1.5-flash`, `gpt-4o-mini` and `Llama3-8B`, representing a diverse set from different model families and sizes. We evaluate them primarily in zero-shot prompting modes. We consider two settings: (i) generating only an answer (A), and (ii) generating an explanation along with the answer (A + E). The latter represents answering the question and then generating an explanation for it. We also analyze few-shot results for the answer-only (A) setting, and evaluate generic CoT (E + A) prompting. More details about each model can be found in Section 5.3.1, and the prompts used are shown in Section 5.3.2.

### 5.3.1 Benchmark Models

We provide details of each model we evaluate on CaT-Bench. For in-context example selection for the LLMs, we utilize the rank_BM25 library[3] available under the Apache 2.0 license.

**`gpt-4o-2024-05-13`** accepts as input any combination of text, audio, image, and video and generates any combination of text, audio, and image outputs. It is especially better at vision and audio understanding compared to existing models.

**`gpt-3.5-turbo`** is an instruction-tuned pretrained language model that powered the original ChatGPT application.

[3] `https://github.com/dorianbrown/rank_bm25`

**gpt-4-turbo-2024-04-09** is the first turbo model in the GPT-4 series. It is more capable and cheaper than GPT-3.5, and supports a 128K context window. It is possibly an 8x220B Mixture-of-Experts model.

**gemini-1.5-flash-latest** is a version of `gemini-1.5-pro` optimized for low latency and inference cost.

**gemini-1.5-pro-latest** is built on a Mixture-of-Experts (MoE) architecture. It is a mid-size multimodal model, optimized for scaling across a wide range of tasks, and also introduces a breakthrough experimental feature in long-context understanding. It is difficult to perform a wide range of experiments with this model due to the imposed rate limits.

**gpt-4o-mini-2024-07-18** has a context window of 128K tokens and supports up to 16K output tokens per request. It surpasses `gpt-4-turbo` and other small models on academic benchmarks across both textual intelligence and multimodal reasoning, and supports the same range of languages as `gpt-4o`.

**gemini-1.0-pro-latest** is built on top of Transformer decoders that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference. They are trained to support 32k context length, employing efficient attention mechanisms such as multi-query attention. Gemini models are trained to accommodate textual input interleaved with a wide variety of audio and visual inputs, and `gemini-1.0-pro` is the mid-sized model in the series.

**claude-3-5-sonnet-20240620** sets new industry benchmarks for graduate-level reasoning (GPQA), undergraduate-level knowledge (MMLU), and coding proficiency (HumanEval).

It shows marked improvement in grasping nuance, humor, and complex instructions, and is exceptional at writing high-quality content with a natural, relatable tone.

**`Meta-Llama3-8B-Instruct`** is a standard decoder-only transformer architecture similar to its predecessor Llama2. Compared to Llama2, `Llama3-8B` uses a tokenizer with a vocabulary of 128K tokens that encodes language much more efficiently, which leads to substantially improved model performance. It also uses grouped query attention (GQA) and was trained on sequences of 8,192 tokens, using a mask to ensure self-attention does not cross document boundaries. `Llama3-8B` is pretrained on over 15T tokens that were all collected from publicly available sources.

### 5.3.2 Prompts Used

We present the different prompts used with the benchmark models in Figure 5.3. All models use the answer-only (A) prompt. All models also share the (A + E) prompt except the Gemini models, which use the NL (A + E) prompt instead. We find that Gemini is better at producing free-form natural language as opposed to a structured code format.

We use a temperature of 0.0 for all experiments with each model to select the most likely token at each step, as this setting allows for reproducibility.

We use the following code snippet to query any OpenAI models.

```
import openai

client = OpenAI(api_key=config["OPENAI_API_KEY"])

response = client.chat.completions.create(
                model=openai_model_name,
                messages=prompt,
                temperature=0.0,
                max_tokens=2,
```

```
            top_p=1.0,
            frequency_penalty=0.0,
            presence_penalty=0.0
        )
```

We use the following code snippet to query any Gemini models.

```
import google.generativeai as genai

genai.configure(api_key=config["GEMINI_API_KEY"])

model = genai.GenerativeModel(args.model_name)
candidatecount, temp, topp, topk = 1, 0.0, 1.0, 1
generation_config = genai.GenerationConfig(
    candidate_count = candidatecount,
    max_output_tokens = args.max_tokens,
    temperature = temp,
    top_p = topp,
    top_k = topk
)
response = model.generate_content(prompt)
```

We run inference on `Llama3-8B` locally on one 40GB Nvidia A6000 GPU using HuggingFace (Wolf et al., 2020).

We use scikit-learn classification report to calculate precision, recall and F1 score per-class and as macro average.

**How Good Are Model Predictions?** Table 5.1 presents the performance of all the models in different settings on Step Order Prediction. We present per-class (DEP and NONDEP) precision, recall and F1 score as well as macro average metrics on the class balanced CAT-BENCH. We make three main observations.

| | | Dep | | | NonDep | | | Macro Avg | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F | P | R | F |
| gpt-3.5-turbo | (A) | 0.62 | 0.50 | 0.55 | 0.58 | **0.69** | 0.63 | 0.60 | 0.60 | 0.59 |
| | (A+E) | 0.56 | 0.71 | 0.63 | 0.61 | 0.45 | 0.52 | 0.58 | 0.58 | 0.57 |
| gpt-4-turbo | (A) | 0.57 | 0.81 | 0.67 | 0.67 | 0.39 | 0.49 | 0.62 | 0.60 | 0.58 |
| | (A+E) | 0.66 | 0.79 | 0.72 | 0.74 | 0.59 | 0.66 | 0.70 | 0.69 | 0.69 |
| gpt-4o | (A) | 0.53 | 0.92 | 0.67 | 0.71 | 0.19 | 0.30 | 0.62 | 0.55 | 0.49 |
| | (A+E) | 0.66 | 0.86 | 0.75 | 0.80 | 0.57 | 0.66 | 0.73 | 0.71 | 0.70 |
| gpt-4o-mini | (A) | 0.53 | 0.88 | 0.66 | 0.64 | 0.22 | 0.33 | 0.59 | 0.55 | 0.50 |
| | (A+E) | 0.62 | 0.78 | 0.69 | 0.70 | 0.52 | 0.59 | 0.66 | 0.65 | 0.64 |
| Llama3-8B | (A) | 0.52 | 0.84 | 0.64 | 0.59 | 0.23 | 0.33 | 0.56 | 0.54 | 0.49 |
| | (A+E) | 0.53 | 0.82 | 0.64 | 0.59 | 0.26 | 0.36 | 0.56 | 0.54 | 0.50 |
| gemini-1.0-pro | (A) | 0.57 | 0.45 | 0.50 | 0.55 | 0.66 | 0.60 | 0.56 | 0.55 | 0.55 |
| | (A+E) | 0.56 | 0.65 | 0.60 | 0.59 | 0.50 | 0.54 | 0.58 | 0.57 | 0.57 |
| gemini-1.5-pro | (A) | 0.55 | 0.77 | 0.64 | 0.61 | 0.37 | 0.46 | 0.58 | 0.57 | 0.55 |
| | (A+E) | **0.67** | 0.93 | **0.78** | 0.89 | 0.54 | **0.67** | **0.78** | **0.74** | **0.73** |
| gemini-1.5-flash | (A) | 0.55 | 0.69 | 0.61 | 0.58 | 0.43 | 0.49 | 0.56 | 0.56 | 0.55 |
| | (A+E) | 0.54 | 0.75 | 0.63 | 0.59 | 0.37 | 0.46 | 0.57 | 0.56 | 0.54 |
| claude-3.5-sonnet | (A) | 0.58 | 0.76 | 0.66 | 0.65 | 0.46 | 0.54 | 0.62 | 0.61 | 0.60 |
| | (A+E) | 0.63 | **0.97** | 0.76 | **0.93** | 0.44 | 0.60 | **0.78** | 0.70 | 0.68 |

Table 5.1: Performance of all models on Step Order Prediction when just providing an answer (A) and when also explaining that answer (A+E). We report per-label as well as macro average precision, recall and F1 score.

| | |
|---|---|
| Answer-only (A) | Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence.<br><br>Goal: {title}<br><br>Procedure:<br>{procedure}<br><br>Must Step {i} happen before Step {j}? Select between yes or no |
| Answer + Explanation (A+E) | Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence.<br><br>Goal: {title}<br><br>Procedure:<br>{procedure}<br><br>1. Must Step {i} happen before Step {j}? Select between yes or no<br>2. Explain why or why not.<br><br>Format your answer as JSON with the key value pairs "binary_answer": "yes/no answer to Q1", "why_answer": "answer to Q2" |
| Explanation + Answer (E+A) | Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence.<br><br>Goal: {title}<br><br>Procedure:<br>{procedure}<br><br>1. Explain why or why not Step {i} must happen {temporal_relation} Step {j}. Think step by step.<br>2. Must Step {i} happen {temporal_relation} Step {j}? Select between yes or no<br><br>Format your answer as JSON with the key value pairs "why_answer": "answer to Q1", "binary_answer": "yes/no answer to Q2" |
| NL Answer + Explanation (A+E) | Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence.<br><br>Goal: {title}<br><br>Procedure:<br>{procedure}<br><br>1. Must Step {i} happen before Step {j}? Select between yes or no<br>2. Explain why or why not.<br><br>Format your answer as follows:<br><br>Answer 1: yes/no<br>Answer 2: your answer in one sentence |

Figure 5.3: Different prompts used for our experiment settings and models. $i$ and $j$ represent step numbers and $temporal\_relation$ can be before/after.

**Models Struggle at Predicting Step Order** In the zero-shot answer-only setting (A), `claude-3.5-sonnet` records the highest F1 score overall of 0.60. `gpt-3.5-turbo` and

gpt-4-turbo are close behind with 0.59 and 0.58 respectively. Surprisingly gpt-4o, the most recent frontier model, fares significantly worse at 0.49 F1. Its less powerful version, gpt-4o-mini, also performs similarly. All three Gemini models (gemini-1.5-pro, gemini-1.0-pro, and gemini-1.5-flash) also only manage an F1 of around 0.55. Llama3-8B also fares poorly with an F1 of 0.49. Most models are comparable or barely better than a random baseline F1 of 0.5 on this balanced dataset showing that they are not able to directly answer the dependence question.

**Generating Explanations Improves Performance** Results for adding explanations to answers is shown in the (A + E) rows in Table 5.1. Seven of the nine models, gpt-3.5-turbo and gemini-1.5-flash being the exceptions, have higher performance when also generating explanations. The biggest improvement in F1 is seen in gpt-4o (+0.21). With explanations, the best result is the 0.73 F1 when using gemini-1.5-pro. While this is substantially better than a random baseline, there is still significant room for improvement.

**Models are Biased Towards Predicting Dependence** Most models exhibit a higher recall for the DEP set and significantly lower recall for the NONDEP set. This is particularly true for the answer only setting ((A) rows), the exceptions being gpt-3.5-turbo and gemini-1.0-pro. Coupled with the substantially lower precision values on the DEP set, this suggests that most models exhibit a bias towards predicting dependence between any given pair of steps. We hypothesize that they use temporal order of steps as a heuristic i.e, if a step appears before another step it is more likely to be dependent than not, and thus becoming biased towards predicting dependencies.

As noted earlier, using explanations improves the overall performance, translating to more balanced precision/recall values on DEP than when predicting answers alone. Since

| | TC | OCC |
|---|---|---|
| gpt-3.5-turbo (A) | 52.39% | 70.42% |
| gpt-3.5-turbo (A+E) | 49.23% | 73.31% |
| gpt-4-turbo (A) | 48.87% | 70.28% |
| gpt-4-turbo (A+E) | 55.00% | 66.97% |
| gpt-4o (A) | **79.86%** | 47.96% |
| gpt-4o (A+E) | 67.46% | 58.17% |
| gpt-4o-mini (A) | 70.56% | 54.79% |
| gpt-4o-mini (A+E) | 57.54% | 56.97% |
| Llama3-8B (A) | 60.42% | **83.87%** |
| Llama3-8B (A+E) | 55.77% | 83.38% |
| gemini-1.0-pro (A) | 53.38% | 73.80% |
| gemini-1.0-pro (A+E) | 49.79% | 66.90% |
| gemini-1.5-pro (A) | 55.14% | 58.24% |
| gemini-1.5-pro (A+E) | 79.65% | 60.21% |
| gemini-1.5-flash (A) | 45.92% | 79.44% |
| gemini-1.5-flash (A+E) | 43.10% | 76.62% |
| claude-3.5-sonnet (A) | 45.14% | 50.21% |
| claude-3.5-sonnet (A+E) | 76.83% | 48.10% |

Table 5.2: Robustness of different models on two consistency metrics, TC and OCC.

the bias towards DEP necessarily means bias against NONDEP, reduction in bias towards DEP also translates to a more balanced performance on both DEP and NONDEP sets. However, even with explanations, the bias towards predicting dependence still remains to some extent for all models. Explanations improve gpt-4o performance the most (+0.36) on NONDEP questions. They do not help smaller models (Llama3-8B) identify dependencies better.

**Temporal Consistency** For a pair of steps $(step_i, step_j)$, the answer to must $step_i$ happen before $step_j$ should be the same as the answer to must $step_j$ happen after $step_i$ regardless of question type. For Table 5.2, we make two main observations: (i) Even the most consistent models gpt-4o, gemini-1.5-pro and claude-3.5-sonnet change their answers to the before and after versions of questions in 20+% of the cases. The rest are far

more inconsistent with `gemini-1.5-flash` changing its answers for more than 55% of the questions; (ii) Surprisingly, adding explanations reduces answer consistency for most models, with `gemini-1.5-pro` (+24%), `gpt-4-turbo` (+14%) and `claude-3.5-sonnet` (+31%) being the only exceptions showing improved consistency upon generating explanations.

**Goal:** lemon zested strawberry shortcakes

---

**Steps:**

...
6. Divide dough in half.
7. Add sugar; beat until stiff peaks form.
8. Place 5cm apart on an ungreased baking tray.

...
12. In mixing bowl, beat cream until it begins to thicken.
13. Gently pat or roll each half into a 1.75cm thick circle.

...

**NonDep Q:** Must Step 12 happen after Step 7?

**NonDep A:** No, adding sugar is a part of making dough but beating the cream makes the filling.

---

**Steps:**

...
6. Divide dough in half.
7. In mixing bowl, beat cream until it begins to thicken.
8. Place 5cm apart on an ungreased baking tray.

...
12. Add sugar; beat until stiff peaks form.
13. Gently pat or roll each half into a 1.75cm thick circle.

...

**NonDep-S Q:** Must Step 12 happen after Step 7?

**NonDep-S A:** No, making the filling with cream can be done in parallel to adding sugar in the dough.

Figure 5.4: Since two steps that are not dependent on each other can be performed in any order, we swap their order in the plan and ask binary questions about them similar to NonDep. While the plan itself is altered, the question remains the same.

**Order Contrastive Consistency** Since step pairs without dependencies can be performed in any order, we introduce a twist on Step Order Prediction in which the step pairs in NonDep

are switched in the plan itself. For each modified plan, we create similar binary questions to NonDep and refer to them as NonDep-S. This helps test whether a model uses the step order as a heuristic to answer the question. The answer to dependency questions about an independent pair of steps should stay the same regardless of the order in which the steps are presented in the plan. Order Contrastive Consistency (OCC) measures the fraction of times models provide consistent answers to the same question across NonDep and NonDep-S. We observe a similar overall inconsistency on OCC as with TC, even from the best models. For most models, generating explanations hurt consistency. Surprisingly, `Llama3-8B` is the most robust according to OCC even though its task performance is lowest. In contrast, `gemini-1.5-pro`, which has the highest task performance, is the least robust as per this metric.

**Chain-of-Thought Struggles** In the experiments thus far, we have asked models to generate explanations for their answers. In contrast, the standard Chain-of-Thought (CoT) prompting strategy (Wei et al., 2022b) can be seen as an explain-then-answer approach (E + A), where we ask the model to generate reasoning or explanation that leads to its answer. In practice, this step-by-step reasoning can be seen as allowing the use of intermediate decoding tokens (like a scratchpad) in service to coming up with a possibly more accurate final answer for many tasks. Table 5.3 compares performance of CoT prompting[4] (E + A) to first predicting the answer and then explaining it (A + E) and simply providing an answer (A), all in the zero-shot setting when using `gpt-4o`.

While chain-of-thought (E + A) results in an improvement over the answer-only setting (A), its performance is far below its counterpart (A + E). This contradicts the expectation that it is better to use CoT for intermediate reasoning rather than answering

[4]We try multiple CoT prompts, all with $temp = 0$, but it has little effect on performance.

| | P | R | F1 | TC % |
|---|---|---|---|---|
| (A) | 0.62 | 0.55 | 0.49 | 79.86 |
| (E+A) | 0.77 | 0.65 | 0.6 | 83.66 |
| (A+E) | 0.73 | 0.71 | 0.7 | 67.46 |

Table 5.3: Performance of `gpt-4o` on the Step Order Prediction task when just predicting the dependency (A) vs predicting and explaining the judgment (A+E) vs using chain-of-thought prompting (E+A).

and then generating explanations. An emerging body of work has demonstrated similar performance drops when using CoT (Liu et al., 2025). However, we note that (E + A) does lead to the highest temporal consistency among all approaches. Looking closer, Table 5.4 shows the performance of these methods on DEP questions. CoT (E + A) has a higher bias towards predicting dependencies compared to both (A) and (A + E).

| | P | R | F1 |
|---|---|---|---|
| (A) | 0.53 | 0.92 | 0.67 |
| (E+A) | 0.59 | 0.98 | 0.74 |
| (A+E) | 0.66 | 0.86 | 0.75 |

Table 5.4: Performance of `gpt-4o` on the Step Order Prediction task when just predicting the dependency (A) vs predicting and explaining the judgment (A+E) vs using chain-of-thought prompting (E+A) on DEP.

Further manual analysis of the explanations shows that (E + A) generates more vacuous explanations (e.g. just stating that a pair of steps are independent without digging deeper into what makes them independent). Additionally, (E + A) makes errors by misunderstanding entity states for a step. Unsurprisingly, both settings suffer from some hallucinations, sometimes missing the content of the steps being asked about. We present an example of such hallucinations in Figure 5.5. These results are further indicators of brittleness and inconsistencies in models' reasoning about step dependencies.

**Goal:** lightly spiced tomato and red lentil soup
**Steps:**
1. Heat the oil in a saucepan.
2. Add onion and cook over low heat for 7-8 mins...
3. Stir in celery and carrot and cook for 3 mins...
...
9. Blend the soup using hand-held blender.
10. Check the consistency.
11. Season to taste.
...
**Question:** Explain why or why not Step 10 must happen after Step 3.
**Model Answer:** Step 10 does not need to happen after Step 3 because the eggplant needs to be cooked before blending.
**Human Score:** 1.0

Figure 5.5: Example of hallucinations produced by GPT-4 in the (E + A) setting.

We also include zero-shot results with `o1-preview`, which was released just before the time of publication of the original paper on this work. `o1-preview` uses search over chain-of-thought explanations as part of its inference process.[5] On CaT-Bench, this model achieves state-of-the-art performance scoring 0.80 F1. This is better than both zero- and few-shot performance of any other model, and even achieves the best TC at 85%. However, even this powerful model shows a bias towards predicting dependence (F1 of 0.83) between steps more than their non-dependence (F1 of 0.76).

**Effect of Improved Prompting Techniques** We also experiment with self-consistency (Mitchell et al., 2022) and few-shot prompting (or in-context learning) (Brown et al., 2020a; Wei et al., 2022a) on `gpt-4o` (A) for Step Order Prediction. Self-consistency decoding attempts to improve accuracy by selecting the most common answer from a set of candidate chain-of-thought completions.

[5] `https://openai.com/index/introducing-openai-o1-preview/`

For self-consistency, we use k=$\{3, 5\}$ and temperature=$\{0.6, 0.8\}$ to sample binary predictions and take the majority of the predicted labels as the model's final answer. Contrary to previous findings (Mitchell et al., 2022), self-consistency did not provide any improvement over the vanilla zero-shot model performance.

| | P | R | F |
|---|---|---|---|
| Zero-shot | 0.62 | 0.55 | 0.49 |
| Self-Consistency | 0.62 | 0.55 | 0.49 |
| Few-shot | 0.79 | 0.70 | 0.68 |

Table 5.5: Performance of different prompting techniques with `gpt-4o` on Step Order Prediction. For self-consistency, we report $k = 3$ and $temp = 0.6$ here, and use 5 exemplars for few-shot experiments.

We use in-context learning (Wei et al., 2022a) with examples selected from the balanced training set using the BM25 (Robertson and Zaragoza, 2009) algorithm. We use k=5 exemplars and dynamically retrieve exemplars from a held-out set that are closest to the test instance. As expected, few-shot prompting improved binary prediction performance substantially (+0.19). In fact, few-shot performance was almost as good as predicting then explaining with `gpt-4o`.

Table 5.6 presents more results for prompting techniques. In particular, we show more variations of self-consistency. We find that varying the temperature and number of samples does not make a significant difference.

### 5.3.3 Understanding Directional Dependencies

We study how models handle questions about different aspects of the same pair of steps. Typically, questions about why a step must happen *before* another require reasoning about preconditions and causes, while answering why a step must happen *after* another requires understanding the effects of any performed actions. Figure 5.6 shows the difference

| | P | R | F | TC | OCC |
|---|---|---|---|---|---|
| Zero-shot | 0.62 | 0.55 | 0.49 | 79.86% | 47.96% |
| Self-Consistency [3, 0.6] | 0.62 | 0.55 | 0.49 | 78.66% | 48.17% |
| Self-Consistency [3, 0.8] | 0.61 | 0.55 | 0.48 | 79.58% | 47.25% |
| Self-Consistency [5, 0.6] | 0.61 | 0.55 | 0.48 | 79.72% | 46.76% |
| Self-Consistency [5, 0.8] | 0.61 | 0.55 | 0.48 | 79.86% | 45.99% |
| Few-shot | 0.79 | 0.70 | 0.68 | 81.48% | 45.56% |

Table 5.6: Accuracy and consistency of different prompting techniques with `gpt-4o` on the Step Order Prediction Task. The first number within the square bracket in self-consistency experiments represents the number of samples and the second represents the temperature at which predictions were sampled. For few-shot experiments, we use k=5 exemplars and use BM25 (Robertson and Zaragoza, 2009) to dynamically retrieve exemplars from a held-out set that are closest to the test instance.

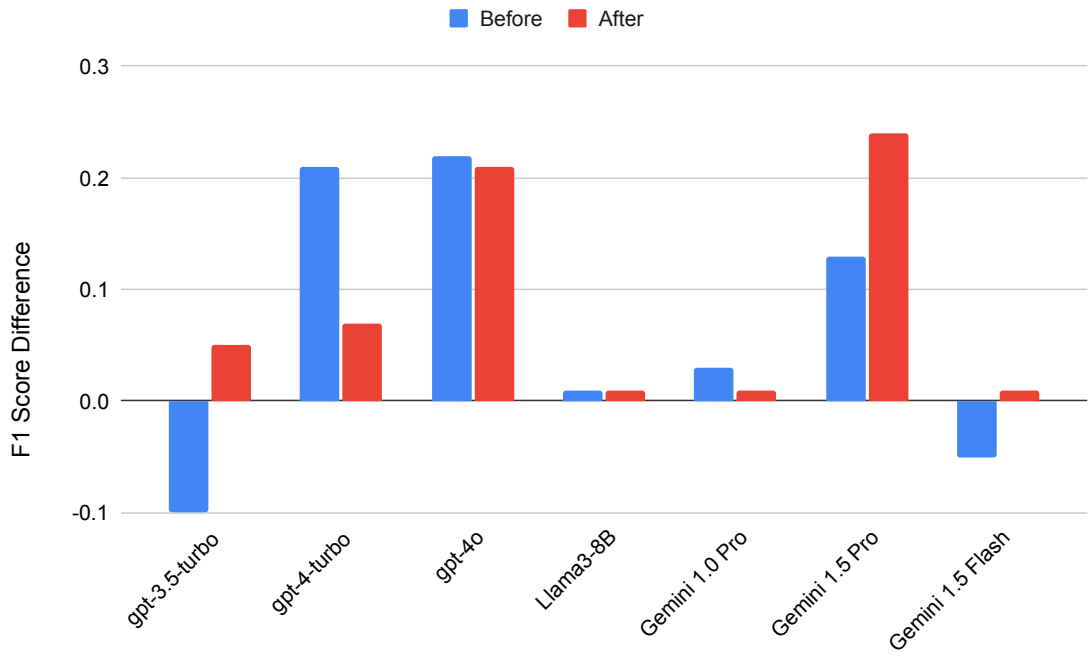



Figure 5.6: Difference in model performance between (A+E) and (A) settings split by temporal relation type (*before* and *after*) asked about in the question. We subtract F1 score in the (A) from the (A+E) setting.

in F1 score between answering and providing an explanation (A + E) and the answer-only setting (A) with different models for these questions. Adding explanations helps all models

understand effects better (*after*). We hypothesize that this is because effects in recipes can be more immediate and hence, would be easier to understand. The biggest gain is seen for `gemini-1.5-pro` and `gpt-4o`, while `Llama3-8B` and `gemini-1.0-pro` do not improve by as much. We note that explanations significantly hurt `gpt-3.5-turbo` and `gemini-1.5-flash` in understanding preconditions, which is unusual.

### 5.3.4 Reasoning as a Function of Step Distance

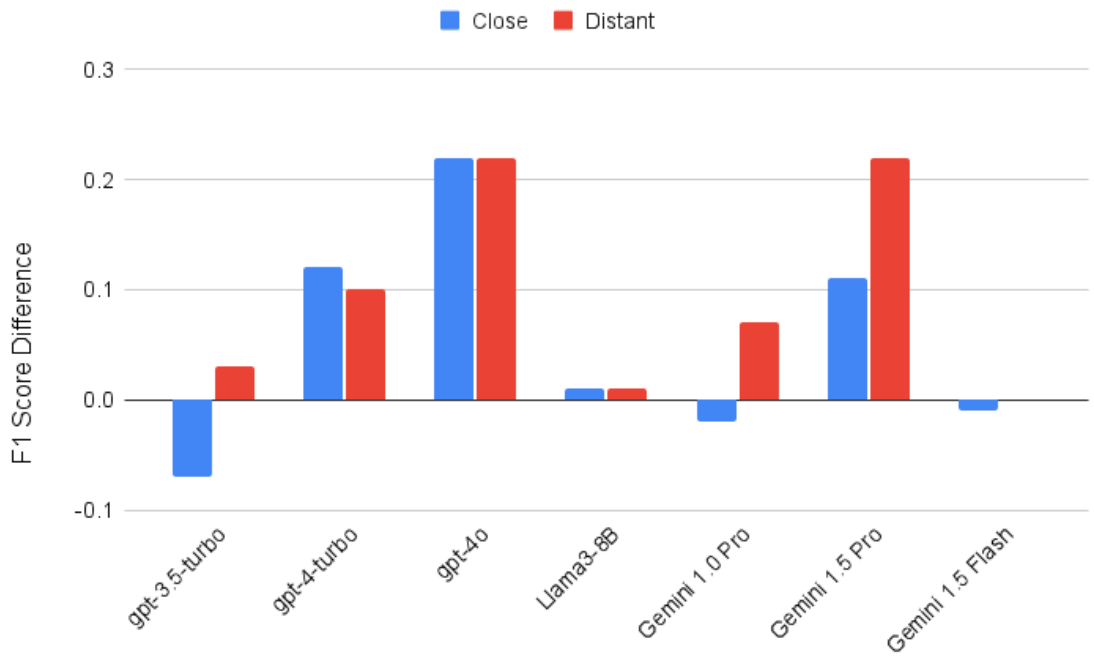



Figure 5.7: Difference in performance of models between (A+E) and (A) settings split by the distance between the steps being asked about in the question.

We study how the distance between the steps in question impacts model performance. A question is said to be about *close* steps $(step_i, step_j)$ if $(j - i) < 3$, and *distant* otherwise. Figure 5.7 shows the difference in F1 score between answering then explaining (A + E) and just answering (A) with different models as a function of step distance. Generating

explanations helps models reason about distant steps, with `gpt-4o` and `gemini-1.5-pro` receiving the greatest benefit. However, they don't help understand dependencies between close steps (which are easier to reason about). In fact, producing explanations even hurts some models, particularly `gpt-3.5-turbo`. We hypothesize that models are likely to predict dependencies between steps that are distant from each other, since it is likely that steps towards the end of a plan depend on ones near the start. We find that this is indeed true: the recall for the nondependent class is very low (usually, $\sim$20%) and models are biased towards predicting a dependency between distant steps.

## 5.4 Human Evaluation of Explanations

For open-ended text generation tasks such as Step Order Explanation, the absence of an automatic metric that correlates well with human judgments is a major challenge (Chen et al., 2019; Ma et al., 2019; Caglayan et al., 2020; Howcroft et al., 2020). We utilize human evaluation with a standardized interface to compare different models. We aim to measure whether a model output is a valid explanation for the given question. We present answers from different models and ask crowd-workers on Amazon Mechanical Turk to assess their correctness. Workers were asked to rate the validity of each answer on a 5-point Likert scale (Likert, 1932) (1 to 5). For each plan, question, and model answer, we ask 3 distinct annotators to provide judgments. An explanation was considered invalid if it does not give a plausible reason that is also relevant to the question.

On a random subset of 480 questions (240 DEP and 240 NONDEP), we conduct a crowdsourced human evaluation of the explanations generated by `gpt-4o`, `gpt-4-turbo`, `gemini-1.5-pro` and `Llama3-8B`, the three best LLMs for Step Order Prediction and an open-source model. Annotators rated how much they agree (1 to 5) with the fact that the answer contains all the relevant details to address what the question requires.

For each explanation, we compute the mean Likert rating from three distinct annotators. First, we report AVG, the overall average of these mean ratings across all 480 instances. To account for cases where the answer is incorrect, we also devised a new metric MODAVG that accounts for cases where the step order prediction is incorrect. To calculate MODAVG, we modify AVG by zeroing out human judgments for explanations where the corresponding prediction is incorrect.

We use weighted Fleiss Kappa to calculate inter-annotator agreement. The weighted agreement score on a 5 point scale was 0.76, indicating high agreement between annotators.

| | AVG | MODAVG |
|---|---|---|
| `Llama3-8B` | 3.26 | 1.87 |
| `gpt-4-turbo` | 3.85 | 2.90 |
| `gpt-4o` | 3.84 | 2.93 |
| `gemini-1.5-pro` | 3.83 | 2.69 |

Table 5.7: Human evaluation metrics for explanations generated by various models in the (A+E) setting.

Table 5.7 presents the quality of model generated explanations as judged by human annotators. As expected, larger models are clearly better than the much smaller `Llama3-8B` on all metrics. There is very little difference between the frontier models, `gpt-4o`, `gpt-4-turbo` and `gemini-1.5-pro`. AVG performance indicates that there is significant room for improvement in the quality of model explanations. On MODAVG, we see that even the best model performance is below 3 ("neither agree nor disagree" with a model's explanation). By this metric, `gemini-1.5-pro` explanations are worse than GPT-4 even though it generates more correct answers. The difference between AVG and MODAVG indicates models are capable of generating convincing explanations for their wrong answers. They produce explanations which justify the opposite of their answer a significant number of times. In fact, `Llama3-8B` does so almost half the time. These results show that models have room for improvement in

their ability and reliability to reason about step dependencies in plans.

### 5.4.1 Human Evaluation Details

Figure 5.8 and Figure 5.9 show the instructions as well as one of the examples presented to annotators when eliciting judgments for model explanations for DEP and NONDEP questions. For each HIT, workers were asked to read the goal of the plan and its steps and then evaluate 6 randomized questions and corresponding answers from models, providing judgments on a Likert scale of 1 to 5. We only selected US-based master turkers who have a minimum lifetime approval rating of 95%. On average, workers took 3 minutes and 51 seconds to judge 6 answers to questions about a plan. We pay them $1.5 per HIT which translates to $23.35 per hour, significantly higher than federal and local minimum wage.

# Instructions:

You will be provided with the goal and instructions of a cooking recipe. Please read the goal and its corresponding set of instructions thoroughly. You will then be asked to evaluate answers to reasoning questions about the recipe. Please use the recipe and your knowledge of the world to judge the answers. Good answers contain all the relevant details required to address a question if it contains a concise and valid explanation, and does not contain any irrelevant information.

The questions ask about the order that steps MUST be done in order to successfully complete the recipe.

**Definitions:**
A recipe step (e.g. Step A) MUST happen after another step (e.g. Step B) if the outcome of Step B is required for completing Step A. For example in a recipe for baking bread, in order to perform the step "Bake the bread at 350 degrees for 30 minutes.", the oven must be preheated first: "Preheat the oven to 350 degrees." must be complete.

Other recipe steps can be done in any order. For example when making a salad "Combine the leaves in a bowl" and "Mix the oil and vinegar together to make salad dressing" can be done in any order.

**Example:**

**Please read the following recipe, questions, and answers.**

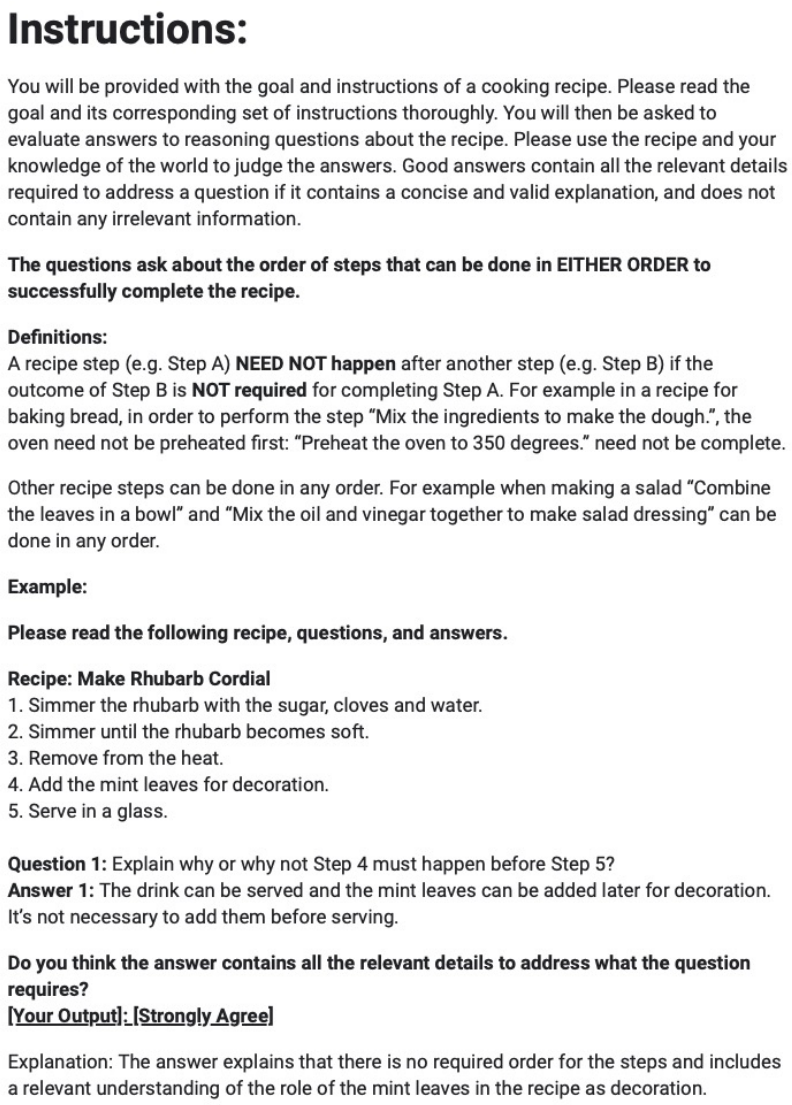
**Recipe: Make Rhubarb Cordial**
1. Simmer the rhubarb with the sugar, cloves and water.
2. Simmer until the rhubarb becomes soft.
3. Remove from the heat.
4. Add the mint leaves for decoration.
5. Serve in a glass.

**Question 1:** Explain why or why not Step 1 must happen before Step 2?
**Answer 1:** The rhubarb should be simmered with sugar, cloves and water to infuse flavor into it till it turns soft which is the next step.

**Does the answer contains all the relevant details to address what the question requires?**
**[Your Output]: 5 / 5 (This is a rating from 1 to 5)**

Explanation: In Step 2 the Rhubarb must be simmered until soft. This will not happen unless it is simmered in water, as in Step 1.

Figure 5.8: Instructions provided to annotators when making judgments about explanations for DEP questions.



# Instructions:

You will be provided with the goal and instructions of a cooking recipe. Please read the goal and its corresponding set of instructions thoroughly. You will then be asked to evaluate answers to reasoning questions about the recipe. Please use the recipe and your knowledge of the world to judge the answers. Good answers contain all the relevant details required to address a question if it contains a concise and valid explanation, and does not contain any irrelevant information.

**The questions ask about the order of steps that can be done in EITHER ORDER to successfully complete the recipe.**

**Definitions:**
A recipe step (e.g. Step A) **NEED NOT happen** after another step (e.g. Step B) if the outcome of Step B is **NOT required** for completing Step A. For example in a recipe for baking bread, in order to perform the step "Mix the ingredients to make the dough.", the oven need not be preheated first: "Preheat the oven to 350 degrees." need not be complete.

Other recipe steps can be done in any order. For example when making a salad "Combine the leaves in a bowl" and "Mix the oil and vinegar together to make salad dressing" can be done in any order.

**Example:**

**Please read the following recipe, questions, and answers.**

**Recipe: Make Rhubarb Cordial**
1. Simmer the rhubarb with the sugar, cloves and water.
2. Simmer until the rhubarb becomes soft.
3. Remove from the heat.
4. Add the mint leaves for decoration.
5. Serve in a glass.

**Question 1:** Explain why or why not Step 4 must happen before Step 5?
**Answer 1:** The drink can be served and the mint leaves can be added later for decoration. It's not necessary to add them before serving.

**Do you think the answer contains all the relevant details to address what the question requires?**
**[Your Output]: [Strongly Agree]**

Explanation: The answer explains that there is no required order for the steps and includes a relevant understanding of the role of the mint leaves in the recipe as decoration.

Figure 5.9: Instructions provided to annotators when making judgments about explanations for NONDEP questions.

### 5.4.2 Additional Human Evaluation Results

We also used two additional metrics to interpret human judgments of model answers. For AVGBIN, we transform each score into a binary value (1 if >3 and 0 otherwise), calculate the mean of these values for an answer and average them over all the data points. We calculate

the majority binary class of judgments for each explanation as MajVote. We report all the metrics for Dep and NonDep in Table 5.8 and Table 5.9 respectively. Looking at AvgBin, we note that there is room for improvement on the quality of model explanations. MajVote indicates that model explanations are convincing even when they are wrong. Note that these metrics do not account for corresponding answer correctness for a model's explanation.

Figure 5.10 presents the distribution of human judgment scores for explanations generated by various models (A + E). We note that models frequently produce high quality answers (5); however, they make too many errors (<3 out of 5) to be consistently reliable.

| | Avg | ModAvg | AvgBin | MajVote |
|---|---|---|---|---|
| Llama3-8B | 3.77 | 3.14 | 0.70 | 75.42 |
| gpt-4-turbo | 3.95 | 3.39 | 0.77 | 79.58 |
| gpt-4o | 4.08 | 3.58 | 0.83 | 87.92 |
| gemini-1.5-pro | 3.98 | 3.8 | 0.77 | 87.08 |

Table 5.8: Human evaluation metrics for explanations generated by various models for Dep questions.

| | Avg | ModAvg | AvgBin | MajVote |
|---|---|---|---|---|
| Llama3-8B | 2.75 | 0.6 | 0.40 | 35.83 |
| gpt-4-turbo | 3.74 | 2.41 | 0.70 | 75.83 |
| gpt-4o | 3.6 | 2.29 | 0.66 | 71.67 |
| gemini-1.5-pro | 3.69 | 1.58 | 0.68 | 75.42 |

Table 5.9: Human evaluation metrics for explanations generated by various models for NonDep questions.

### 5.4.3 Inter-annotator Agreement

We measure the inter-rater reliability of annotators' judgments using weighted Fleiss's Kappa (Marasini et al., 2016), following the weighting scheme used by Bastan et al. (2020). This measure has a penalty for each dissimilar rating based on the distance between the two ratings. For instance, if two annotators classify a document as positive, the agreement weight

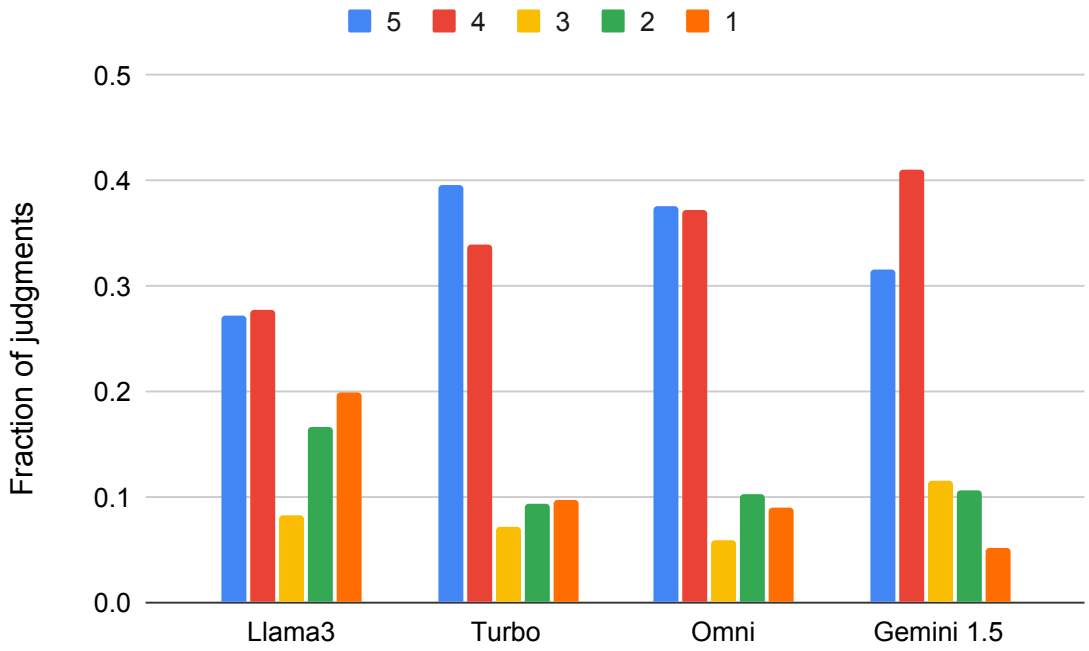



Figure 5.10: Distribution of human judgment scores for explanations generated by various models (A+E).

is 1, but if one classifies it as positive and the other as slightly positive, the agreement weight is lower. The weights between different classes are shown in Table 5.10 where negative, slightly negative, neutral, slightly positive, and positive classes are shown with -2, -1, 0, 1, and 2. We follow the setup used in Bastan et al. (2020) for a similar multi-class labeling task.

| | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -2 | 1 | cos $\pi/8$ | cos $\pi/4$ | cos $3\pi/8$ | 0 |
| -1 | cos $\pi/8$ | 1 | cos $\pi/8$ | cos $\pi/4$ | cos $3\pi/8$ |
| 0 | cos $\pi/4$ | cos $\pi/8$ | 1 | cos $\pi/8$ | cos $\pi/4$ |
| 1 | cos $3\pi/8$ | cos $\pi/4$ | cos $\pi/8$ | 1 | cos $\pi/8$ |
| 2 | 0 | cos $3\pi/8$ | cos $\pi/4$ | cos $\pi/8$ | 1 |

Table 5.10: Inter-class weights used for computing inter-annotator agreement

Table 5.11 presents the inter-annotator agreement for judgments on answers to

different types of questions in CaT-Bench. The high Fleiss Kappa values demonstrate strong agreement between annotators and indicate reliability of our human evaluation framework.

| | Weighted Fleiss Kappa | Weighted Binarized Fleiss Kappa |
|---|---|---|
| Dep | 0.808 | 0.941 |
| NonDep | 0.705 | 0.934 |

Table 5.11: Inter-annotator agreement as measured by Fleiss Kappa for each question type in CaT-Bench

## 5.5 Error Analysis

**Goal:** vanilla rye cheesecake
**Steps:**
1. Preheat the oven to 150 degrees C
2. Place the biscuits into a plastic bag and crush.
...
10. Place baking dish onto bottom rack of preheated oven and fill halfway with boiling water.
...
**Question:** Explain why or why not Step 10 must happen after Step 2.
**Model Answer:** Step 10 must happen after Step 2 because the baking dish with boiling water needs to be placed in the oven just before baking the cheesecake to create a moist environment, which is necessary after the biscuit base has been prepared and the filling has been added.
**Human Score:** 5.0

---

**Goal:** lemon zested strawberry shortcakes
**Steps:**
1. In bowl, combine strawberries and sugar...
2. In bowl, combine flour, sugar, baking powder and salt.
3. Rub butter into flour until forming coarse crumbs.
...
9. Bake at 200 C / Gas 6 for 8-10 minutes.
10. Remove to a wire rack; cool for 15 minutes.
11. In bowl, combine butter and lemon zest; set aside.
...
**Question:** Explain why or why not Step 10 must happen after Step 2.
**Model Answer:** Step 10 involves cooling the baked shortcakes, which are prepared and baked in steps 5 through 9, and does not depend on the flour mixture preparation in step 2.
**Human Score:** 1.0

Figure 5.11: Examples of cases where GPT-4 comes up with good (upper box) and bad (lower box) answers. This error is of the multi-hop dependency type. To make shortcakes, removing the cake from the oven (Step 10) is dependent on baking the cake (step 9) which is later dependent on combining the ingredients (Step 2).

To better understand model failures, we sample and analyze 50 explanations generated by `gpt-4o` (A + E) where it produces an incorrect answer. We identified 4 major types of errors:

- Multi-hop dependency (40%): Failure to understand that two steps might be related

through an intermediate step. For instance, to make shortcakes in Figure 5.11, removing the cake from the oven (Step 10) is dependent on baking the cake (Step 9) which, consequently, is dependent on combining the ingredients (Step 2). The model does not seem to understand the transitive nature of such dependencies.

- Effects (20%): Failure to understand that an effect of the preceding step leads to the succeeding step, e.g., serving a cake must happen after mixing ingredients and consequently baking. LLMs fail to identify additive effects of steps in a plan which enable a later step, leading to goal completion.
- Preconditions (18%): Failure to understand a condition that needs to be satisfied for a step to happen. For instance, to add sauce in a later step, meatballs need to be cooked first so they can be added to the sauce.
- Irrelevant Answers (18%): Model produces answers that are unrelated to the step being asked about. It is surprising to see that LLMs mistakenly produce an answer about an unrelated step, particularly given that the input context is short (well below maximum context length) and can be easily used for grounding.

**Goal:** fusilli and meatballs
**Steps:** ...
15. Cook the meatballs until browned on all sides.
16. Drain on kitchen paper.
17. Add the meatballs to the tomato sauce.
...
20. Toss with 3 ladlefuls of sauce in serving bowl..
...
**Question:** Explain why or why not Step 20 must happen after Step 15.
**Model Answer:** Step 20 can happen after Step 19 as it involves tossing the cooked pasta with the sauce, which is independent of browning the meatballs in Step 15.

---

**Goal:** chocolate cake
**Steps:**
1. Preheat oven to 180C / Gas Mark 4.
2. Put chocolate chips, coffee, water in bowl.
3. Microwave until melted.
...
7. Gradually beat in sugar till fluffy.
8. Gradually add and beat in egg yolks.
...
**Question:** Explain why or why not Step 3 must happen before Step 7.
**Model Answer:** The chocolate mixture needs to be cooled before it is added to the batter in Step 10.

---

**Goal:** almond and apple cake
**Steps:**...
7. Add half the flour and half the milk.
8. Use a wooden spoon to stir until well combined.
...
13. Set aside to cool slightly before plating.
14. Serve warm.
...
**Question:** Explain why or why not Step 14 must happen after Step 8.
**Model Answer:** Step 14 is the final step and cannot happen after mixing the ingredients.

Figure 5.12: Examples of types of GPT-4 errors. The top box contains an error related to misunderstanding preconditions, the second one about producing irrelevant answers and the last one about misunderstanding effects.

## 5.6 Limitations and Ethical Considerations

While our work only considers cooking recipes as procedural texts, our methods can in principle be applied to many other domains. Medical practice guidelines, repair manuals, and software tutorials among others are domains worth investigating. Our work only investigates English-language documents and this limits the generalizability of our findings to other languages.

We benchmark a reasonably diverse set of LLMs, covering 3 model families and models of varying sizes. Due to the current fast-paced landscape of LLM development, future work should continue to evaluate more LLMs on CaT-Bench. It is also difficult for any one person to adequately evaluate the various aspects of plans, particularly recipes. To alleviate this problem, we use 3 crowd-sourced annotators to judge model explanations and consider their average judgment, but recognize the limitations of this solution. We do show high inter-annotator agreement using Weighted Fleiss Kappa (Marasini et al., 2016), demonstrating the reliability of our results.

We use BM25 as a reasonable choice to find similar exemplars. We acknowledge that there are more modern techniques for selecting in-context examples, but this step is not the focus of our current work. We leave further exploration of exemplar selection methods to future work. For a domain like recipe text where texts are long and less amenable to a single embedding vector approach, keyword-based retrieval such as BM25 is very effective. Since there are no gold explanations, we cannot combine few-shot prompting and chain-of-thought (or answer-then-explain) settings for `gpt-4o`. We also do not advocate for using gold explanations along with automatic metrics to judge model explanations due to established inadequacies in using automatic metrics for free-form generations. Due to strict rate limits on the recently released Gemini models, we are unable to analyze `gemini-1.5-pro` through chain-of-thought and other prompting techniques. For consistency, we analyze `gpt-4o` since

it has similar performance (A + E) on CaT-Bench.

Prior work has shown that LLMs exhibit various types of bias. While they do not generate free-form language for our binary prediction task, it is possible, though highly unlikely, that biases explicitly come up in the explanations. Deploying such unreliable models into critical infrastructure and relying on them for decisions can cause harm to users.

## 5.7 Conclusions

Understanding plans requires reasoning about their different aspects such as preconditions and effects. We introduce CaT-Bench, a new benchmark to evaluate the causal and temporal reasoning abilities about plans. Despite the remarkable strength of current SOTA LLMs, we found that none of them are very good at understanding whether one step in a plan must precede (or succeed) another. Particularly, they are much worse at knowing when there is *not* a dependency between steps. We also found that LLM predictions are not robust as measured by two metrics of consistency. Prompting LLMs to provide an answer and then to explain it improves performance significantly, and surprisingly performs better than chain-of-thought prompting. Human evaluation of these explanations shows that models have a long way to go at understanding dependencies.

## 5.8 Author Contributions

This work was done jointly with co-first author Yash Kumar Lal while he was a PhD student at Stony Brook University. Vanya Cohen led the work on the automated evaluation, selecting the Recipe Flow Graph Corpus (Yamakata et al., 2020) as the benchmark corpus and defining the step order prediction task used for evaluating model action reasoning. Yash Kumar Lal, building on his thesis work on explanation-based reasoning, led the work on the explanation-reasoning aspects of the chapter, defining the human evaluation experiments that assess model-generated explanations of step dependencies. Vanya Cohen designed the annotation interface and ran the crowdsourced human evaluation on Amazon Mechanical Turk.

## Acknowledgments

This material is based on research that is supported in part by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003 and in part by the National Science Foundation under the award IIS #2007290. This material is also based upon work supported by DARPA's Perceptually-enabled Task Guidance (PTG) program under Contract No. HR001122C0007.

# Chapter 6: MET-Bench: Multimodal Entity Tracking for Evaluating the Limitations of Vision-Language and Reasoning Models

*Presented at the International Conference on Machine Learning (ICML) 2026.*

## 6.1 Introduction

Action reasoning and forward planning require more than understanding the precondition relationships between actions evaluated in Chapter 5. They require world modeling (Ding et al., 2026): understanding how the state of entities in the world evolves through sequences of actions. The corrective planner in Chapter 3 provides this capability, simulating entity states forward through a planning domain to identify and repair precondition violations. For a model to perform such reasoning without an external planner, it must be able to track entity states internally. Given the growing reliance on foundation models as decision-making agents (Cohen et al., 2024), it is important to evaluate whether they can perform entity state tracking on their own rather than relying on external symbolic planners. Moreover, many important applications of entity state tracking, including robotic manipulation (Yoneda et al., 2024; NVIDIA et al., 2025), video question-answering (He et al., 2025), and computer-control agents (Bishop et al., 2024; Sager et al., 2025), require reasoning over visual inputs, making multimodal state tracking a necessary capability. This chapter evaluates whether current vision-language models possess this capability. In Cohen and Mooney (2026), we introduce MET-Bench, a multimodal benchmark that tests whether models can maintain a consistent representation of entity states as those states are updated by sequences of textual and visual observations.

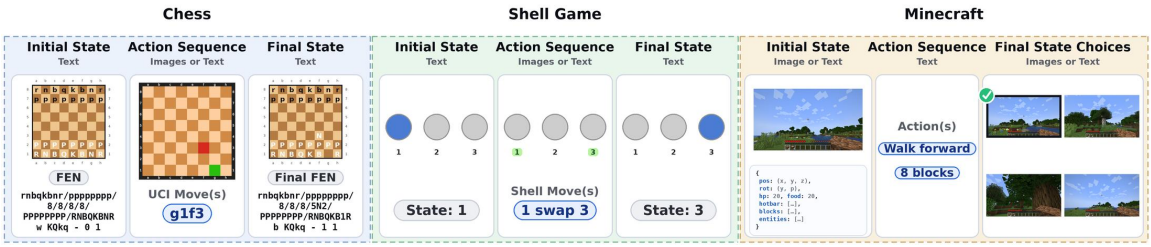



Figure 6.1: In the multimodal entity tracking benchmark (MET-Bench), a vision-language model (VLM) predicts the final entity state from the initial state and actions that update the entity state. In Chess and Shell Game, the initial entity state is provided as text and the actions are given as images or text. The predicted final state is a text representation of the entity state. In Minecraft, the initial state is given as text or image and the actions as text. The task is to predict the correct final state from a multiple choice candidate set of text or image representations.

Early progress on entity tracking emerged within text-only tasks. From coreference resolution (Hobbs, 1978; Lappin and Leass, 1994) and discourse processing to narrative comprehension, computational linguistics has long grappled with the challenge of maintaining coherent entity representations across textual contexts (Bunescu and Paşca, 2006; Schank and Abelson, 1977b).

While significant progress has been made in text-only tasks (Liu et al., 2023b; Papalampidi et al., 2022), the broader challenge of understanding how entities change through sequences of actions or events remains an open challenge (Fagnou et al., 2024; Kim and Schuster, 2023; Toshniwal et al., 2022). This limitation becomes apparent in tasks requiring integration of information across multiple modalities, an increasingly important frontier in AI as systems are asked to reason about content that combines text with other modalities like images and video.

Our work examines this challenge through the lens of multimodal entity state tracking, where changes to entity states must be understood from both textual descriptions and visual observations. This setting provides a natural extension to classical NLP problems while also

connecting to emerging research in multimodal dialogue, robotics, and human-AI interaction. We focus specifically on scenarios where multimodal models must reason about world-state changes described through a combination of text and images.

To systematically evaluate models' capabilities in this multimodal setting, we study three domains: *multimodal Chess*, *Shell Game*, and *Minecraft*. Through Chess and Shell Game, we assess how effectively current language models can track entity states when updates are conveyed through both text and images. Our analysis reveals substantial disparities in how models process text-based and image-based entity-state updates, highlighting fundamental limitations in their multimodal understanding. While Chess and the Shell Game provide structured testbeds for evaluating entity tracking, real-world applications often involve more ambiguous and dynamic environments. However, by isolating state-tracking performance in controlled settings, we establish a clear baseline for assessing multimodal reasoning, one that provides a straightforward means of evaluation where difficulty is easily controlled.

We also evaluate state prediction performance in Minecraft, which offers a setting more applicable to real-world state tracking performance, while still enabling controlled evaluation. In Minecraft, we instead vary the state modality to analyze the difference in models' abilities to reason about states presented as visual and textual inputs.

## 6.2 Background

We formulate the problem of *multimodal entity tracking* as a sequential state estimation task, where an agent must infer the final state of a system given an initial state and a series of observed actions. Formally, at each time step $t$, the environment is in state $\mathbf{S}_t$, and transitions occur according to an action sequence $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T)$. The agent receives $\mathbf{a}_t$ corresponding to each action, which may be textual $\mathbf{a}_t^{\text{text}}$ or visual $\mathbf{a}_t^{\text{image}}$. The

objective is to infer the final state:

$$\mathbf{S}_T = \mathrm{f}(\mathbf{S}_0, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T),$$

where $\mathrm{f}$ is the function modeling entity state updates.

In Chess and Shell Game, MET-Bench represents the initial and final states of each domain as text but also evaluates models' ability to track entity state changes through images. This approach isolates the multimodal entity tracking challenge by ensuring that models begin and end with well-defined textual representations while processing state transitions visually. This assesses the models' capacity to maintain coherent entity representations across modalities while minimizing confounding errors from perceptual failures, which remain a limitation of current vision-language models (Sharma et al., 2024). In Minecraft, actions are represented as text and the input and final states are represented either as text or images.

### 6.2.1 Chess

Chess is a well-studied domain for testing entity tracking of deep learning models (Toshniwal et al., 2022). The state $S_t$ represents an $8\times8$ board configuration expressed in Forsyth-Edwards Notation (FEN), actions correspond to legal chess moves from real games, and action observations consist of either symbolic Universal Chess Interface (UCI) notation or visual board-image descriptions of moves. We likewise adopt chess as an entity tracking testbed where the task is to maintain a correct representation of the board state across a sequence of moves.

Utilizing real games from Millionbase[1] (Toshniwal et al., 2022), we generate sequences of states and actions (moves) using standard chess notation: UCI for actions and

[1] rebel13.nl/rebel13

FEN for board states. **Text-Encoded Moves** (each action is provided as a short UCI textual description, e.g., "`e2e4`" for moving a piece from e2 to e4); **Image-Encoded Moves** each action is accompanied by a rendered image serving as a visual representation of the move; see Fig. 6.1. The visual action depictions were created to enable high perceptual accuracy, as shown by the results in Table 6.7. In both cases, the final output is the FEN-encoded location of each piece on the board after a sequence of moves. The dataset includes multiple game trajectories of varying length, capturing a variety of piece types and board states.

### 6.2.2 Shell Game

Shell Game is a classic demonstration of hidden-state tracking. A ball is placed under one of three cups (or shells), which are then swapped pairwise in succession. The goal is to track which cup currently hides the ball. The state $S_t$ tracks the hidden position of a ball under three shells, and actions correspond to swaps between pairs of shells. Other works have explored text-based shell-game-like domains with varying levels of added complexity (Li et al., 2021; Long et al., 2016; Kim and Schuster, 2023).

Shell Game has a simpler entity-state and action space than Chess. However, while many frontier models have been trained on UCI/FEN-encoded chess games, the Shell Game is, to the best of our knowledge, not present in the training data of these models, though there may be analogous tasks in the pretraining data.

We simulate repeated Shell Game swaps to create a set of Shell Game trajectories. Swap actions are either: **Text-Encoded Swaps** "`x swap y`", where $\{x, y\} \subset \{1, 2, 3\}$; **Image-Encoded Swaps** an image depicting the shells being swapped, with the ball not visible; see Fig. 6.1. The ground truth entity state after the game finishes is a single number indicating the final shell position of the ball.

### 6.2.3 Minecraft

Minecraft is a more complex domain where state tracking requires reasoning about partial observation, dynamic environments, and visually complex scenes. The state $S_t$ is the player's local first-person world state, either as an image or textual description including position, orientation, nearby blocks, and visible scene contents. Actions correspond to low-level player inputs such as walking, turning, attacking, using an item, climbing, or descending (e.g., "`Walk forward 8 blocks, attack`"). Unlike Chess and Shell Game, where the model predicts a symbolic final state, in Minecraft the model must predict the correct next state from four choices, sampled from adjacent parts of the trajectory. The ground truth output is a single multiple-choice index indicating which candidate is the correct next state.

We collect a dataset of trajectories across multi-step scripted behaviors, including `collect wood`, `explore`, and `build structure`. Each trajectory consists of rendered first-person frames paired with per-tick game telemetry in text. We generate benchmark examples by sampling an input state, the input controls used, and four candidate next states, one of which is the true successor. We evaluate two parallel state encodings: **Text-Encoded States** where the input and candidate states are rendered from game telemetry as structured textual descriptions and **Image-Encoded States** where the input and candidate states are first-person Minecraft screenshots; see Fig. 6.1. The text-encoded states contain rich task information, which the model must infer from the visual observations.

The text setting serves as an upper bound on task performance: the state information is structured, so accuracy reflects the model's state reasoning ability alone. The image setting presents similar information visually, so any performance gap directly quantifies the deficit introduced by visual state understanding. If a model possessed equal competence in visual and textual reasoning, the two settings would yield comparable accuracy. The

magnitude of this gap therefore isolates the contribution of visual understanding to overall entity tracking failure.

## 6.3 Approach

We utilize the standard chat-based input schema exposed by current models that consists of interleaved user-provided and model-generated messages. Figure 6.2 shows the prompting strategy used for the Chess domain. Similarly, Figure 6.3 details the prompting strategy used for the Shell Game domain.

In the case of image-action input, the text actions are replaced with their image-rendered versions and a text description of how to interpret the image-actions is provided. Fig. 6.1 shows the image representations used for the Chess and Shell Game actions. These image representations were created through visual prompt engineering to maximize the classification accuracy of actions depicted. Various common image representations were explored including arrows, bounding boxes, and symbolic markers. The image depiction of the game actions is explained to the language model every time images are provided using the prompts shown in Figures 6.4 & 6.5.

| Role | Messages |
|---|---|
| **User** | You are a helpful assistant that tracks chess moves in a game and produces the final FEN. The initial state is:<br>rnbqkbnr/pppppppp/8/8/8/8/ PPPPPPPP/RNBQKBNR w KQkq - 0 1<br>Here are the moves played:<br>e2e4<br>e7e5<br>Now what is the final FEN? Output FINAL ANSWER: [FEN]. |
| **Assistant** | FINAL ANSWER: rnbqkbnr/pppp1ppp/8/4p3/4P3 /8/PPPP1PPP/RNBQKBNR w KQkq - 0 2 |

Figure 6.2: An example zero-shot user-assistant exchange in the **Chess** domain, showing the initial board state as FEN, two UCI moves (e2e4, e7e5) and the final state. For image actions, the UCI moves are replaced with their visual representations and a description of how to interpret these images. The FEN is line-broken for readability.

| Role | Messages |
|---|---|
| **User** | The shell game is a classic game where a ball is hidden under one of three shells. You are a helpful assistant that tracks the position of the ball. The ball starts under shell 2. Here are the moves played:<br>1 swap 3<br>2 swap 3<br>Now what is the final position of the ball? Only output the number 1, 2, or 3. |
| **Assistant** | 3 |

Figure 6.3: An example zero-shot user-assistant exchange in the **Shell Game** domain, illustrating how the system tracks swaps to determine the ball's final shell.

| Role | Messages |
|---|---|
| **User** | `You are a helpful assistant that interprets image-based actions in chess.`<br>`Here is an image representing a move:`<br>**[Image Input]**<br>`In UCI notation, what move does the arrow on the chessboard represent? The move is from the green square to the red square. (e.g., 'e2e4'). Only output the move and nothing else.` |
| **Assistant** | `e2e4` |

Figure 6.4: An example user-assistant exchange in the **Chess** domain, where the assistant identifies the move represented in the image.

| Role | Messages |
|---|---|
| **User** | `You are a helpful assistant that interprets image-based actions in the shell game.`<br>`Here is an image representing a swap:`<br>**[Image Input]**<br>`In shell game notation, which shells are being swapped in the image? Shells are labeled '1', '2', '3' and the shells being swapped have their numbers highlighted in green. Only output a pair like '1 swap 3' and nothing else.` |
| **Assistant** | `1 swap 3` |

Figure 6.5: An example user-assistant exchange in the **Shell Game** domain, where the assistant identifies the shell swap represented in the image.

### 6.3.1 Benchmark Statistics

Table 6.1 shows the train/validation/test split sizes for each published domain in MET-Bench. Results are reported on a $500$ example subset of each test set and $100$ examples for the largest reasoning models.

Table 6.1: Split sizes for the publication datasets. Metrics are reported on a 500 or 100 example subset of the test sets as discussed in Section 6.4.2.

| Split | Examples |
|---|---|
| **Chess** | |
| Train | 99,000 |
| Validation | 3,000 |
| Test | 3,000 |
| **Shell Game** | |
| Train | 50,000 |
| Validation | 1,000 |
| Test | 5,000 |
| **Minecraft** | |
| Train | 5,000 |
| Validation | 500 |
| Test | 1,500 |

### 6.3.2 GRPO Training for Entity Tracking

To investigate whether reinforcement learning can improve entity tracking in open-weight vision-language models, we apply GRPO to train models on MET-Bench tasks.

**Reward Function Design** For entity tracking, we use reward functions that leverage the automatic verifiability of our benchmark tasks. For Chess, the reward is the accuracy of the predicted board:

$$R_{\text{chess}}(y, y^*) = \frac{1}{64} \sum_{i=1}^{64} \mathbb{1}[y_i = y_i^*],$$

For Shell Game, the reward is binary based on the ball position:

$$R_{\text{shell}}(y, y^*) = \mathbb{1}[y = y^*].$$

We train Gemma 3 4B IT (GemmaTeam et al., 2025). For the Chess and image settings at 10 actions, the base policies do not output valid chain-of-thought; to enable

RL training, we initialize the policy with a valid output format by finetuning on synthetic demonstrations. Separately, we find that finetuning on only the final entity state does not generalize, supporting the results in Tables 6.4 and 6.6 that sequential state tracking is made easier by generating intermediate reasoning.

Table 6.2 summarizes the key hyperparameters used for GRPO training across domains and modalities. All experiments are implemented using the TRL library (von Werra et al., 2020) with the Adafactor optimizer (Shazeer and Stern, 2018). Training is performed on a subset of $10{,}000$ 10-action trajectories until convergence. For the Chess domain and image modality, the Gemma 3 4B IT base policy is not strong enough to output valid reasoning traces. For this setting, we initialize the policy with finetuning on $1{,}000$ synthetic demonstrations.

Table 6.2: GRPO training hyperparameters.

| Param | Gemma 3 4B IT | | Gemma 3 4B IT | |
|---|---|---|---|---|
| | Chess (Text) | Shell (Text) | Chess (Image) | Shell (Image) |
| Optimizer | Adafactor | Adafactor | Adafactor | Adafactor |
| Learning Rate | 5e-7 | 1e-7 | 5e-7 | 5e-7 |
| Batch Size | 4 | 4 | 1 | 1 |
| Grad. Accum. | 16 | 16 | 16 | 16 |
| Group Size | 8 | 8 | 8 | 8 |

## 6.4 Experiments

Our experiments test a wide range of models (Section 6.4.1) and settings to evaluate different aspects of frontier-model entity-tracking performance. For all experiments, the models are sampled with a temperature of zero.

### 6.4.1 Model Details

Table 6.3 details the models used for evaluation and training. Models were selected to sample a wide range of leading providers, including open-source and closed models, and various sizes and protocols for reasoning training. We report separate results for models with dedicated reasoning training or decoding strategies beyond chain-of-thought output.

Table 6.3: Models used in MET-Bench evaluation and training.

| Model Name |
|---|
| *Anthropic* |
| Claude 3.7 Sonnet (Anthropic, 2025) |
| Claude 4.5 Opus (Anthropic, 2025) |
| Claude Sonnet 4.6 (Anthropic, 2026) |
| *Google DeepMind* |
| Gemini 2.5 Flash, Pro (Gemini Team, 2025) |
| Gemma 3 4B IT, 27B IT (GemmaTeam et al., 2025) |
| *Meta* |
| Llama 4 Maverick (Meta, 2025) |
| *MiniMax* |
| MiniMax-VL-01 (MiniMax et al., 2025) |
| *Mistral* |
| Mistral Small 3.2 24B (Mistral, 2025) |
| *OpenAI* |
| GPT-4o, 4o mini (OpenAI et al., 2024a) |
| GPT-4.1, 4.1-mini, 4.1-nano (OpenAI, 2025a) |
| GPT-5.2, 5.4-mini (Singh et al., 2025) |
| o1 (OpenAI et al., 2024b) |
| o3, o4-mini (OpenAI, 2025b) |
| *Qwen* |
| Qwen3-VL 8B, 30B, 235B-A22B (Qwen Team, 2025) |

### 6.4.2 Tracking in Text Outperforms Images

We evaluate the difference in accuracy when tracking entity states from text and image actions in zero-shot, few-shot, chain-of-thought, and reasoning settings and report 95% confidence intervals (CI). For the few-shot experiments, $N = 5$ examples are selected at random from the training set and prepended to the test example. To evaluate the effect of chain-of-thought reasoning, we prompt the model as in the zero-shot case but append "think step by step before producing a final answer." In both domains, we evaluate on a set of $500$ examples selected at random from the test set, each with a sequence length of ten actions, and $100$ for reasoning models.

Table 6.4: Entity tracking accuracy (95% CI) in Chess for text-only and image-only actions. In the Few-Shot setting $N = 5$ in-context examples are used. Methods with explicit reasoning perform best. The baseline is predicting the Game Start board configuration.

| Model | Chess | |
|---|---|---|
| | Text | Image |
| **Baseline** | | |
| Game Start | 74.9 ± 0.5 | 74.9 ± 0.5 |
| **Zero-Shot** | | |
| GPT-4o | 91.6 ± 0.3 | 76.0 ± 0.5 |
| GPT-4o-mini | 75.6 ± 0.5 | 55.3 ± 0.5 |
| GPT-4.1 | 85.6 ± 0.4 | 67.7 ± 0.5 |
| GPT-4.1-mini | 82.5 ± 0.4 | 77.5 ± 0.5 |
| GPT-4.1-nano | 77.5 ± 0.5 | 73.3 ± 0.5 |
| Gemini-2.5-Flash | 91.0 ± 0.3 | 66.9 ± 0.5 |
| LLaMA-4 Maverick | 86.7 ± 0.4 | 51.1 ± 1.2 |
| MiniMax-VL-01 | 85.9 ± 0.4 | 73.4 ± 0.5 |
| Claude 3.7 Sonnet | 96.1 ± 0.2 | 70.2 ± 0.5 |
| **Few-Shot (n=5)** | | |
| GPT-4o | 94.3 ± 0.2 | 77.6 ± 0.5 |
| GPT-4o-mini | 74.5 ± 0.5 | 74.2 ± 0.5 |
| GPT-4.1 | 86.6 ± 0.4 | 64.9 ± 0.5 |
| GPT-4.1-mini | 81.2 ± 0.4 | 76.9 ± 0.5 |
| GPT-4.1-nano | 74.5 ± 0.5 | 74.6 ± 0.5 |
| Gemini-2.5-Flash | 91.3 ± 0.3 | 72.0 ± 0.5 |
| LLaMA-4 Maverick | 85.8 ± 0.4 | 48.1 ± 0.6 |
| MiniMax-VL-01 | 88.0 ± 0.8 | 40.8 ± 1.2 |
| Claude 3.7 Sonnet | 99.2 ± 0.1 | 77.7 ± 0.5 |
| **Chain-of-Thought** | | |
| GPT-4o | 94.2 ± 0.3 | 83.4 ± 0.4 |
| GPT-4o-mini | 67.0 ± 0.5 | 41.8 ± 0.5 |
| GPT-4.1 | 98.1 ± 0.1 | 75.3 ± 0.5 |
| GPT-4.1-mini | 86.4 ± 0.4 | 77.6 ± 0.5 |
| GPT-4.1-nano | 49.0 ± 0.6 | 34.6 ± 0.5 |
| Gemini-2.5-Flash | 77.0 ± 1.0 | 44.6 ± 1.2 |
| LLaMA-4 Maverick | 82.4 ± 0.4 | 61.9 ± 0.5 |
| MiniMax-VL-01 | 62.3 ± 0.5 | 32.8 ± 0.5 |
| Claude 3.7 Sonnet | 99.5 ± 0.1 | 96.2 ± 0.2 |
| **Reasoning** | | |
| o1 | 98.2 ± 0.3 | 83.5 ± 0.9 |
| o3 | **99.9 ± 0.1** | 45.5 ± 1.2 |
| o4-mini | 84.2 ± 0.9 | 78.0 ± 1.0 |
| Gemini-2.5-Pro | 77.4 ± 1.0 | 76.8 ± 1.0 |
| Gemini-2.5-Flash | 40.2 ± 1.2 | 17.4 ± 0.9 |
| Claude 3.7 Sonnet | **99.8 ± 0.1** | 96.0 ± 0.5 |
| Claude 4.5 Opus | **99.9 ± 0.1** | **99.5 ± 0.2** |
| GPT-5.2 | 98.0 ± 0.4 | **99.3 ± 0.2** |

**Chess** In the Chess domain, we report the per-square accuracy of the predicted board, that is the ratio of correctly predicted pieces (or absence of a piece) to the total number of board tiles. The 'Game Start' baseline is the accuracy of predicting the initial board configuration. After only ten actions, much of the board remains unchanged, so this is a strong baseline.

Table 6.4 reports the accuracy for text and image actions in the chess domain. Performance in the text modality is significantly better than in the image modality. In the zero-shot setting, the best-performing text model Claude 3.7 Sonnet achieves an impressive

$96.1\% \pm 0.2\%$ accuracy, while its image-based counterpart drops sharply to $70.2\% \pm 0.5\%$. A similar trend holds across models, indicating that current models can perform entity tracking in text but fail to integrate visual updates as effectively.

Few-shot prompting, chain-of-thought prompting, and reasoning lead to performance improvements. Because predicting the initial board position is a strong baseline for the accuracy metric, we report additional metrics (exact FEN match, changed square accuracy, and board legality) to assess the robustness of model predictions. We consider three complementary measures that provide a more fine-grained picture of model performance. **Exact-match FEN accuracy** requires the model to predict the exact FEN representation, offering the strictest metric on board prediction. **Changed-square accuracy** isolates performance on squares whose contents differ from the starting position, testing whether models genuinely track game dynamics rather than memorizing the initial board layout. **Legal board position** is the fraction of predictions that are legal board positions. Together, these metrics complement per-square accuracy to provide a robust view of entity tracking performance in Chess. These more challenging evaluation settings confirm that the strongest models do track entity state, and that high performance is not an artifact of per-square accuracy. For non-reasoning models like GPT-4o, in many cases the per-square accuracy is high even when these less permissive metrics are low, indicating the models have not mastered the task.

We also perform an ablation using a random mixture of text and image-specified moves in Section 6.4.7. As expected, performance drops smoothly as the proportion of image-actions increases.

Table 6.5: Chess entity tracking with stronger metrics. **Exact**: exact-match FEN accuracy (%). **Changed**: accuracy on squares that changed from the initial position (%). **Legal**: fraction of predictions that are legal board positions (%). All values report mean ± 95% CI.

| | Text-Only | | | Image-Only | | |
|---|---|---|---|---|---|---|
| Model | Exact | Changed | Legal | Exact | Changed | Legal |
| **Baseline** | | | | | | |
| Game Start | 0.0 ± 0.0 | 0.0 ± 0.0 | 100.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 100.0 ± 0.0 |
| **Zero-Shot** | | | | | | |
| GPT-4o | 2.0 ± 1.3 | 81.7 ± 0.8 | 58.6 ± 4.3 | 0.0 ± 0.4 | 14.1 ± 0.8 | 95.4 ± 1.9 |
| GPT-4o-mini | 0.0 ± 0.4 | 25.2 ± 0.9 | 24.2 ± 3.7 | 0.0 ± 0.4 | 33.9 ± 1.0 | 21.8 ± 3.6 |
| GPT-4.1 | 1.2 ± 1.0 | 78.4 ± 0.9 | 66.4 ± 4.1 | 0.0 ± 0.4 | 25.8 ± 1.0 | 74.6 ± 3.8 |
| GPT-4.1-mini | 0.4 ± 0.7 | 63.5 ± 1.1 | 35.8 ± 4.2 | 0.0 ± 0.4 | 15.4 ± 0.8 | 58.0 ± 4.3 |
| GPT-4.1-nano | 0.0 ± 0.4 | 46.1 ± 1.1 | 51.0 ± 4.4 | 0.0 ± 0.4 | 3.4 ± 0.4 | 92.2 ± 2.4 |
| Gemini-2.5-Flash | 2.4 ± 1.4 | 83.6 ± 0.8 | 38.8 ± 4.3 | 0.0 ± 0.4 | 7.2 ± 0.6 | 59.6 ± 4.3 |
| LLaMA-4 Maverick | 0.0 ± 0.4 | 75.6 ± 0.9 | 40.4 ± 4.3 | 0.0 ± 0.4 | 48.2 ± 1.1 | 1.0 ± 0.9 |
| MiniMax-VL-01 | 0.2 ± 0.5 | 67.9 ± 1.0 | 77.0 ± 3.7 | 0.0 ± 0.4 | 2.6 ± 0.4 | 94.0 ± 2.1 |
| Claude 3.7 Sonnet | 18.8 ± 3.4 | 90.0 ± 0.7 | 79.6 ± 3.5 | 0.0 ± 0.4 | 30.7 ± 1.0 | 82.2 ± 3.3 |
| **Few-Shot (n=5)** | | | | | | |
| GPT-4o | 3.2 ± 1.6 | 89.7 ± 0.7 | 83.6 ± 3.2 | 0.2 ± 0.5 | 33.5 ± 1.0 | 99.6 ± 0.7 |
| GPT-4o-mini | 0.2 ± 0.5 | 47.1 ± 1.1 | 58.6 ± 4.3 | 0.0 ± 0.4 | 5.6 ± 0.5 | 63.0 ± 4.2 |
| GPT-4.1 | 2.4 ± 1.4 | 81.7 ± 0.8 | 79.6 ± 3.5 | 0.0 ± 0.4 | 29.6 ± 1.0 | 74.4 ± 3.8 |
| GPT-4.1-mini | 0.6 ± 0.8 | 66.1 ± 1.0 | 70.4 ± 4.0 | 0.2 ± 0.5 | 34.6 ± 1.0 | 74.8 ± 3.8 |
| GPT-4.1-nano | 0.0 ± 0.4 | 48.3 ± 1.1 | 50.4 ± 4.4 | 0.0 ± 0.4 | 0.7 ± 0.2 | 98.6 ± 1.1 |
| Gemini-2.5-Flash | 26.4 ± 3.9 | 90.2 ± 0.7 | 82.0 ± 3.4 | 0.2 ± 0.5 | 27.0 ± 1.0 | 89.0 ± 2.7 |
| LLaMA-4 Maverick | 0.4 ± 0.7 | 80.8 ± 0.9 | 66.8 ± 4.1 | 0.0 ± 0.4 | 36.0 ± 1.1 | 10.2 ± 2.7 |
| MiniMax-VL-01 | 0.0 ± 0.4 | 68.7 ± 1.0 | 71.2 ± 4.0 | 0.0 ± 1.8 | 16.7 ± 1.8 | 54.0 ± 9.6 |
| Claude 3.7 Sonnet | 47.6 ± 4.4 | 98.5 ± 0.3 | 96.4 ± 1.7 | 0.2 ± 0.5 | 36.1 ± 1.1 | 75.6 ± 3.8 |
| **Chain-of-Thought** | | | | | | |
| GPT-4o | 7.8 ± 2.4 | 87.0 ± 0.7 | 85.0 ± 3.1 | 2.0 ± 1.3 | 61.4 ± 1.1 | 82.4 ± 3.3 |
| GPT-4o-mini | 0.0 ± 0.4 | 11.3 ± 0.7 | 5.0 ± 1.9 | 0.0 ± 0.4 | 18.2 ± 0.8 | 11.6 ± 2.8 |
| GPT-4.1 | 42.8 ± 4.3 | 71.0 ± 1.0 | 69.0 ± 4.0 | 1.0 ± 0.9 | 24.3 ± 0.9 | 26.4 ± 3.9 |
| GPT-4.1-mini | 5.0 ± 1.9 | 76.8 ± 0.9 | 54.2 ± 4.4 | 0.8 ± 0.9 | 56.1 ± 1.1 | 30.2 ± 4.0 |
| GPT-4.1-nano | 0.0 ± 0.4 | 30.6 ± 1.0 | 32.2 ± 4.1 | 0.0 ± 0.4 | 3.8 ± 0.4 | 10.6 ± 2.7 |
| Gemini-2.5-Flash | 18.4 ± 3.4 | 69.9 ± 1.0 | 46.8 ± 4.4 | 5.2 ± 2.0 | 65.8 ± 1.0 | 34.2 ± 4.1 |
| LLaMA-4 Maverick | 10.6 ± 2.7 | 82.1 ± 0.8 | 64.2 ± 4.2 | 0.0 ± 0.4 | 33.6 ± 1.0 | 28.4 ± 3.9 |
| MiniMax-VL-01 | 0.0 ± 0.4 | 52.7 ± 1.1 | 49.8 ± 4.4 | 0.0 ± 0.4 | 8.5 ± 0.6 | 26.2 ± 3.8 |
| Claude 3.7 Sonnet | 81.2 ± 3.4 | 98.9 ± 0.2 | 99.2 ± 0.9 | 30.0 ± 4.0 | 92.0 ± 0.6 | 97.0 ± 1.5 |
| **Reasoning** | | | | | | |
| o1 | 61.0 ± 9.4 | 96.2 ± 0.9 | 90.0 ± 6.0 | 15.0 ± 7.0 | 62.0 ± 2.3 | 77.0 ± 8.2 |
| o3 | 99.0 ± 2.6 | 100.0 ± 0.1 | 99.0 ± 2.6 | 12.0 ± 6.4 | 36.6 ± 2.3 | 50.0 ± 9.6 |
| o4-mini | 32.0 ± 9.0 | 82.4 ± 1.8 | 34.0 ± 9.1 | 6.0 ± 4.8 | 64.5 ± 2.3 | 37.0 ± 9.3 |
| Gemini-2.5-Pro | 39.0 ± 9.4 | 75.5 ± 2.1 | 67.0 ± 9.1 | 49.0 ± 9.6 | 74.9 ± 2.1 | 69.0 ± 8.9 |
| Gemini-2.5-Flash | 17.0 ± 7.3 | 66.6 ± 2.3 | 52.0 ± 9.6 | 9.0 ± 5.7 | 55.6 ± 2.4 | 33.0 ± 9.1 |
| Claude 3.7 Sonnet | 88.0 ± 6.4 | 99.5 ± 0.4 | 99.0 ± 2.6 | 52.0 ± 9.6 | 93.5 ± 1.2 | 95.0 ± 4.5 |
| Claude 4.5 Opus | 97.0 ± 3.7 | 100.0 ± 0.1 | 100.0 ± 1.8 | 89.0 ± 6.2 | 99.0 ± 0.5 | 100.0 ± 1.8 |
| GPT-5.2 | 95.0 ± 4.5 | 98.0 ± 0.7 | 98.0 ± 3.2 | 88.0 ± 6.4 | 98.5 ± 0.6 | 100.0 ± 1.8 |

Table 6.6: Entity tracking accuracy (95% CI) in Shell Game for text-only and image-only actions. In the Few-Shot setting $N = 5$ in-context examples are used. Methods with explicit reasoning perform best.

| Model | Shell | |
|---|---|---|
| | Text | Image |
| **Baseline** | | |
| Random | 33.3 | 33.3 |
| **Zero-Shot** | | |
| GPT-4o | 33.0 ± 4.1 | 32.2 ± 4.1 |
| GPT-4o-mini | 33.6 ± 4.1 | 32.4 ± 4.1 |
| GPT-4.1 | 32.2 ± 4.1 | 36.4 ± 4.2 |
| GPT-4.1-mini | 31.4 ± 4.1 | 30.8 ± 4.0 |
| GPT-4.1-nano | 31.2 ± 4.0 | 30.8 ± 4.0 |
| Gemini-2.5-Flash | 35.0 ± 4.2 | 37.0 ± 4.2 |
| LLaMA-4 Maverick | 30.8 ± 4.0 | 33.2 ± 4.1 |
| MiniMax-VL-01 | 30.8 ± 4.0 | 31.2 ± 4.0 |
| Claude 3.7 Sonnet | 35.4 ± 4.2 | 37.8 ± 4.2 |
| **Few-Shot (n=5)** | | |
| GPT-4o | 33.6 ± 4.1 | 32.0 ± 4.1 |
| GPT-4o-mini | 34.8 ± 4.2 | 32.2 ± 4.1 |
| GPT-4.1 | 34.6 ± 4.2 | 33.0 ± 4.1 |
| GPT-4.1-mini | 36.4 ± 4.2 | 32.8 ± 4.1 |
| GPT-4.1-nano | 36.6 ± 4.2 | 36.4 ± 4.2 |
| Gemini-2.5-Flash | 31.4 ± 4.1 | 35.2 ± 4.2 |
| LLaMA-4 Maverick | 35.4 ± 4.2 | 35.2 ± 4.2 |
| MiniMax-VL-01 | 36.4 ± 4.2 | 32.0 ± 4.1 |
| Claude 3.7 Sonnet | 32.4 ± 4.1 | 36.0 ± 4.2 |
| **Chain-of-Thought** | | |
| GPT-4o | 98.2 ± 1.2 | 36.6 ± 4.2 |
| GPT-4o-mini | 66.2 ± 4.1 | 33.8 ± 4.1 |
| GPT-4.1 | **99.8 ± 0.5** | 37.6 ± 4.2 |
| GPT-4.1-mini | **100.0 ± 0.4** | 72.0 ± 3.9 |
| GPT-4.1-nano | 61.8 ± 4.2 | 32.2 ± 4.1 |
| Gemini-2.5-Flash | 94.0 ± 4.8 | 34.0 ± 9.1 |
| LLaMA-4 Maverick | 77.0 ± 3.7 | 34.6 ± 4.2 |
| MiniMax-VL-01 | 77.4 ± 3.7 | 34.4 ± 4.2 |
| Claude 3.7 Sonnet | **100.0 ± 0.4** | 77.4 ± 3.7 |
| **Reasoning** | | |
| o1 | **100.0 ± 1.9** | 92.6 ± 2.3 |
| o3 | **100.0 ± 1.9** | 63.0 ± 9.3 |
| o4-mini | **100.0 ± 1.9** | 33.0 ± 4.1 |
| Gemini-2.5-Pro | **100.0 ± 1.9** | **100.0 ± 1.9** |
| Gemini-2.5-Flash | **100.0 ± 1.9** | 42.0 ± 4.1 |
| Claude 3.7 Sonnet | **100.0 ± 1.9** | 87.0 ± 6.6 |
| Claude 4.5 Opus | **100.0 ± 1.9** | **100.0 ± 1.9** |
| GPT-5.2 | **100.0 ± 1.9** | **99.0 ± 2.6** |

**Shell Game** The final state is a single number $n \in \{1, 2, 3\}$ that gives the position of the ball, and we measure the accuracy of predicting it. The naive baseline picks a position uniformly at random.

Table 6.6 reports the accuracy for text and image actions in Shell Game. The results follow a pattern similar to that of Chess, with text-based tracking outperforming image-based tracking. However, unlike in Chess, the performance in the zero-shot and few-shot settings is close to random.

Chain-of-thought and reasoning lead to large performance increases and multiple models attain near-perfect accuracy in text. These results suggest that when guided to decompose the task step-by-step, models can reason more effectively (but still largely imperfectly) using image inputs, a finding that complements the results of performing cascaded inference as discussed below.

As in Chess, we perform an ablation using a mixture of text and image-specified moves in Section 6.4.7. Unlike in Chess, Shell Game performance drops as the action-modality mixture approaches $50\%$ and then recovers as the proportion of image actions reaches $100\%$, but to a lower accuracy than text-only tracking. Models may struggle more with reasoning over a mixture of modalities in some cases.

### 6.4.3 Reasoning Improves Long Sequence Accuracy

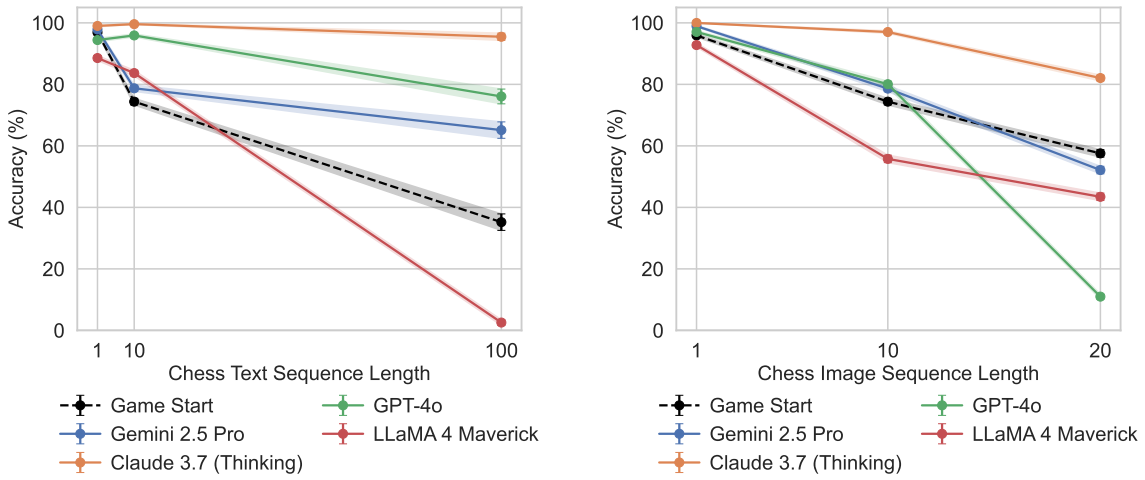
We vary the sequence length of the actions to quantify the effect of compounding errors on the models. These sequences range from one to $100$ actions in the text action modality and one to $20$ in the image action modality. This serves to quantify the relative drop-off in performance among models in the reasoning and chain-of-thought settings.

**Chess** Figure 6.6a plots model accuracy against increasing sequence lengths of text actions. The models are evaluated in the chain-of-thought setting for sequence lengths of one to $100$ text actions. Figure 6.6b plots model accuracy against increasing sequence lengths of image actions. The models are evaluated in the chain-of-thought setting for sequence lengths of one to $20$ image actions. Reasoning models like Claude 3.7 Sonnet Thinking are able to handle longer sequence lengths with a smaller decrease in accuracy. In contrast, the non-reasoning models experience sharp decreases in accuracy after only a few actions.

**Shell Game** In the text modality in Figure 6.7a, the reasoning models Gemini 2.5 Pro and Claude 3.7 Sonnet Thinking attain the highest accuracies. Gemini 2.5 Pro performs

nearly perfectly at a sequence length of 100 actions, where the non-reasoning models' performance is significantly degraded. In the image modality results in Figure 6.7b, Gemini 2.5 Pro performs better than the other models, which see a rapid decrease in accuracy with sequence lengths longer than ten actions. By 20 actions, the performance of all models except Gemini 2.5 Pro has degraded to random guessing.

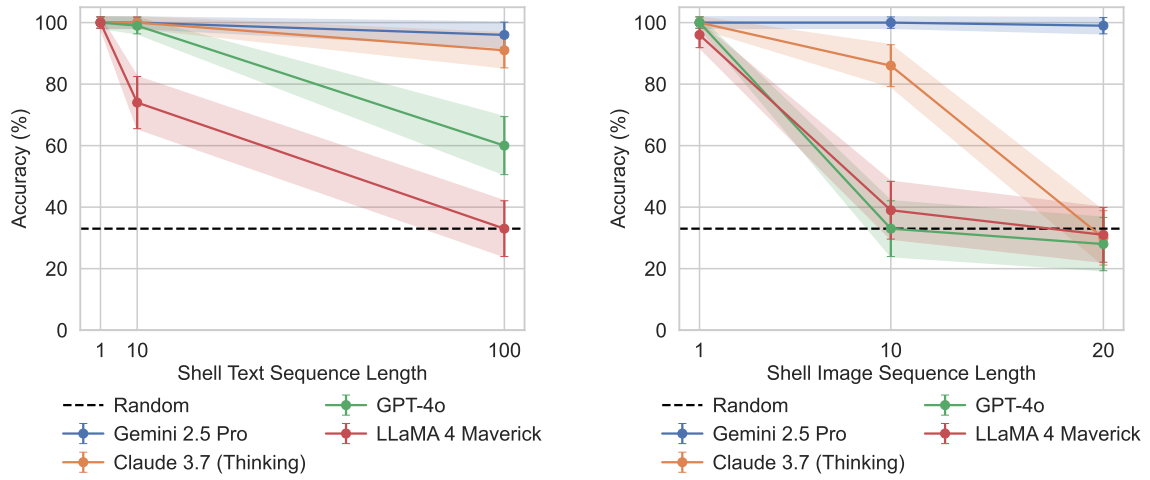
The superior performance of reasoning models suggests that structured inference mechanisms, such as test-time search or chain-of-thought, are crucial for maintaining coherent entity states over long sequences. This aligns with prior findings that models trained on structured reasoning tasks (e.g., mathematics, coding) develop stronger implicit state-tracking capabilities (Kim et al., 2024), whereas standard vision-language models struggle to track entities beyond short sequences.




(a) Chess accuracy with text actions.

(b) Chess accuracy with image actions.

Figure 6.6: In the text action setting, the reasoning model Claude 3.7 Sonnet Thinking and GPT-4o maintain the highest accuracy at longer action-sequence lengths. All models struggle to maintain accurate board representations in the image action setting, with Claude 3.7 Sonnet Thinking performing the best. 95% confidence intervals.




(a) Shell Game accuracy with text actions.

(b) Shell Game accuracy with image actions.

Figure 6.7: In the text action setting, the reasoning models Claude 3.7 Sonnet Thinking and Gemini 2.5 Pro achieve the highest performance over long action sequences. In the image action setting the accuracy of all models except Gemini 2.5 Pro decreases to random guessing by 20 actions. 95% confidence intervals.

### 6.4.4 Tracking Primarily Limited by Visual Reasoning

Using the text actions predicted from the images in the image-action classification task, we devise a cascaded approach to test the effect of performing multimodal inference in two separate steps. Given a sequence of image-based observations $\boldsymbol{a}_t^{\text{image}}$, a vision-language model (VLM) first predicts the corresponding text-based action sequence $\hat{\boldsymbol{a}}^{\text{text}} = g(\boldsymbol{a}^{\text{image}})$, where $g$ maps images to text using the same prompt from image evaluation; Figure 6.5. The text actions are then used for chain-of-thought inference to estimate the final state as $\boldsymbol{S_T} = f(\boldsymbol{S_0}, \hat{\boldsymbol{a}}^{\text{text}})$. This removes the need for the model to perform entity tracking directly in the image modality, isolating the effect of perceptual failures.

To differentiate perception errors from visual-reasoning errors, we evaluate cascaded inference. Table 6.7 shows the performance of models on predicting the text action represented by each image. While some models fail to accurately perceive the Chess

moves, they perceive the Shell Game actions with perfect accuracy. The performance in this cascaded setting is more similar to the text-action performance, showing that the model has the task-knowledge needed to perform entity tracking in both domains, but cannot reason effectively directly from the image modality. Together, this indicates that perception of the image-actions is not the only limiting factor for effective entity tracking with image inputs, though there is some error contribution from perception error in Chess.

Table 6.7: Entity tracking and perception accuracy ($95\%$ CI) for Chess and Shell Game. Cascaded inference generally recovers the performance of the text setting, showing the main limitation is visual reasoning not perception. **Perception** accuracy is image-action classification accuracy. In **Cascaded** inference, the model first maps each image action to the text representation of the action, then performs tracking as in **Text**. $\Delta$ is **Cascaded** minus **Image**.

| | Chess | | | | | Shell Game | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Perception | Text | Image | Cascaded | $\Delta$ | Perception | Text | Image | Cascaded | $\Delta$ |
| GPT-4o | $92.9 \pm 0.7$ | $94.2 \pm 0.3$ | $83.4 \pm 0.4$ | $92.8 \pm 0.3$ | +9.4 | $100.0 \pm 0.1$ | $98.2 \pm 1.2$ | $36.6 \pm 4.2$ | $98.2 \pm 1.2$ | +61.6 |
| GPT-4o-mini | $44.7 \pm 1.4$ | $67.0 \pm 0.5$ | $41.8 \pm 0.5$ | $64.3 \pm 0.5$ | +22.5 | $100.0 \pm 0.1$ | $66.2 \pm 4.1$ | $33.8 \pm 4.1$ | $68.8 \pm 4.0$ | +35.0 |
| Gemma 3 27B IT | $71.3 \pm 1.3$ | $53.7 \pm 0.6$ | $48.6 \pm 0.6$ | $52.8 \pm 0.6$ | +4.2 | $100.0 \pm 0.1$ | $56.8 \pm 4.3$ | $30.4 \pm 4.0$ | $58.4 \pm 4.3$ | +28.0 |
| Mistral Small 3.2 24B | $63.9 \pm 1.3$ | $49.5 \pm 0.6$ | $16.3 \pm 0.4$ | $37.4 \pm 0.5$ | +21.1 | $100.0 \pm 0.1$ | $97.4 \pm 1.4$ | $35.2 \pm 4.2$ | $97.2 \pm 1.5$ | +62.0 |
| Qwen3-VL 235B-A22B | $99.8 \pm 0.2$ | $72.1 \pm 0.5$ | $63.0 \pm 4.2$ | $70.3 \pm 0.5$ | +7.4 | $100.0 \pm 0.1$ | $100.0 \pm 0.4$ | $38.8 \pm 4.3$ | $100.0 \pm 0.4$ | +61.2 |
| GPT-5.4-mini | $99.8 \pm 0.2$ | $92.6 \pm 0.3$ | $88.7 \pm 0.3$ | $93.6 \pm 0.3$ | +4.9 | $100.0 \pm 0.1$ | $94.8 \pm 2.0$ | $32.4 \pm 4.1$ | $91.8 \pm 2.4$ | +59.4 |

### 6.4.5 Minecraft Multimodal World Modeling

Table 6.8 evaluates the complementary task of Minecraft game state prediction. The model receives a visual or textual game state and action and must predict from four candidates the next state. This task is complementary to Chess and Shell Game in that the multimodal variation lies in the *state representation* rather than the action representation. By measuring the performance difference between text-based tasks and the related but more challenging image-based tasks, we can understand the extent to which visual state representations limit model reasoning compared to their textual equivalents in a more naturalistic domain.

The results in Table 6.8 confirm the pattern observed in the structured domains:

models are more accurate on text-based inputs than on image-based inputs. Claude Sonnet 4.6 has the highest performance in both settings, reaching $88.4\% \pm 2.8\%$ with text states and $41.8\% \pm 4.3\%$ with image states. Notably, even the best-performing model in the image setting only exceeds the random baseline by a modest margin, underscoring the difficulty of reasoning about state transitions from first-person visual observations.

Scaling trends are evident in text but not image. Among the Qwen3 VL family, accuracy improves with model size in the text setting, rising from $49.4\%$ at 8B parameters to $73.0\%$ at 235B. However, the corresponding gains in the image setting are far more modest, suggesting that simply scaling model parameters provides diminishing returns for visual state reasoning.

Table 6.8: Minecraft state-prediction accuracy (95% CI) using chain-of-thought prompting. To complement the other domains, Minecraft evaluates state prediction from **Text State:** serialized game-state observations and **Image State:** first-person rendered views. Both settings use text actions and the answer is selected from four candidates. Each reported model cell is evaluated on $500$ examples; the random baseline is $25\%$.

| Model | Minecraft | |
|---|---|---|
| | **Text State** | **Image State** |
| **Baseline** | | |
| Random | 25.0 | 25.0 |
| **Chain-of-Thought** | | |
| Gemma-3 4B IT | $39.4 \pm 4.3$ | $26.6 \pm 3.9$ |
| Gemma-3 27B IT | $51.6 \pm 4.4$ | $27.8 \pm 3.9$ |
| Qwen3-VL 8B | $49.4 \pm 4.4$ | $28.4 \pm 3.9$ |
| Qwen3-VL 30B | $72.0 \pm 3.9$ | $31.4 \pm 4.1$ |
| Qwen3-VL 235B | $73.0 \pm 3.9$ | $34.2 \pm 4.1$ |
| GPT-4o | $71.0 \pm 4.0$ | $37.0 \pm 4.2$ |
| Claude Sonnet 4.6 | $\mathbf{88.4 \pm 2.8}$ | $\mathbf{41.8 \pm 4.3}$ |

### 6.4.6 GRPO Improves Entity Tracking

As shown in Tables 6.4 and 6.6, methods that use chain-of-thought and explicit reasoning training and inference perform better than single-step zero-shot and few-shot methods. RL training is one way to improve the reasoning abilities of models. We train open-source VLMs to improve entity tracking on the 10-action subset of MET-Bench. We observe limited cross-modal transfer, where improvements in one modality do not result in improvements in the other modality. This suggests models may process entity tracking in text and image through different pathways. Robust multimodal entity tracking may require architectural innovations or training strategies beyond what RL provides.

Table 6.9 shows that GRPO training yields substantial gains in the trained modality. For example, when trained on text-only Shell Game inputs, Gemma 3 4B IT improves from $37.6\%$ (near random) to $92.2\%$ accuracy. Despite the improvement in single-modality entity tracking, RL reasoning training does not transfer robustly to the held-out modality. In Shell Game, the accuracy on the held-out modality does not improve. In Chess, there is improvement from the model learning to produce more valid FENs, but only to the level predicting the starting game board. This finding suggests that the entity tracking capabilities learned through RL training are modality-specific.

Table 6.9: Entity tracking accuracy (%) after GRPO training for Gemma 3 4B IT on the 10-action setting. GPT-4o-mini and Gemma 3 27B IT are shown for comparison. 95% confidence intervals.

| Model | Text | Image |
|---|---|---|
| **Shell Game** | | |
| Gemma 3 4B IT (Base) | $37.6 \pm 4.2$ | $32.8 \pm 4.1$ |
| + GRPO Text | $92.2 \pm 2.4$ | $27.4 \pm 3.9$ |
| + GRPO Image | $39.4 \pm 4.3$ | $90.4 \pm 2.6$ |
| GPT-4o-mini | $66.2 \pm 4.1$ | $33.8 \pm 4.1$ |
| Gemma 3 27B IT | $56.8 \pm 4.3$ | $30.4 \pm 4.0$ |
| **Chess** | | |
| Gemma 3 4B IT (Base) | $45.4 \pm 0.6$ | $35.9 \pm 0.5$ |
| + GRPO Text | $97.1 \pm 0.2$ | $73.3 \pm 0.5$ |
| + GRPO Image | $67.9 \pm 0.5$ | $92.5 \pm 0.3$ |
| GPT-4o-mini | $67.0 \pm 0.5$ | $41.8 \pm 0.5$ |
| Gemma 3 27B IT | $53.7 \pm 0.6$ | $48.6 \pm 0.6$ |

### 6.4.7 Models Struggle to Integrate Mixed Modalities

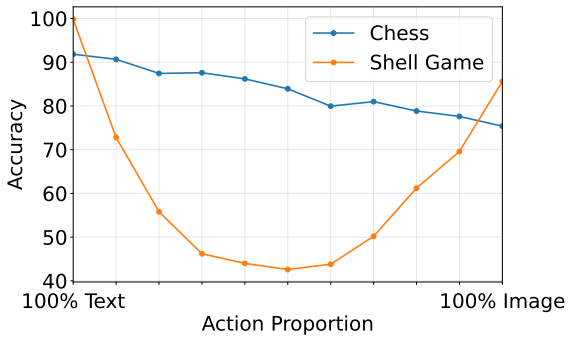



Figure 6.8: Chain-of-thought entity tracking accuracy for Chess and Shell Game with GPT-4o. The data splits range from 100% text-encoded actions to 100% image-encoded actions. The plot illustrates the change in accuracy as actions shift between modalities.

While MET-Bench focuses on parallel, multimodal state tracking tasks, in this ablation we seek to understand how well models can reason about mixed-modality updates. We evaluate performance as we vary the proportion of text and image-based action representations. As shown in Figure 6.8, for Chess performance degrades smoothly as the fraction of image actions increases, rather than exhibiting an abrupt collapse. However, for Shell Game, the mixture of text and image actions is challenging for the model to reason over.

## 6.5 Discussion

We demonstrate a significant performance gap between text-based and image-based entity tracking across many state-of-the-art vision-language-reasoning models. This disparity persists across Chess, Shell Game, and Minecraft tasks, suggesting a fundamental limitation in current architectures' ability to reason about entity states through visual observations. This finding is surprising given the results showing models can accurately perceive and classify individual visual actions in Shell Game, and to a lesser extent in Chess. When models first translate visual inputs to text before performing entity tracking, they achieve performance comparable to pure text-based tracking. This indicates that the models possess the relevant task knowledge and reasoning capabilities, but struggle to apply them directly in the visual domain.

Likewise, the results in Minecraft show that VLMs achieve substantially higher performance on state prediction when the world state is represented as text rather than as a first-person visual observation. Because the text condition provides complete and unambiguous state information, it serves as an upper bound on task performance, reflecting textual reasoning ability alone. The large gap between text and image accuracy and limited improvements from model scaling indicate that visual state understanding remains far from equivalent to textual reasoning.

The effectiveness of chain-of-thought prompting, particularly in the Shell Game domain where it improved Claude 3.7 Sonnet's accuracy from near random to $100.0\%$ for text and $77.4\%$ for images, highlights the importance of explicit reasoning for entity tracking. Current models can perform complex entity tracking when guided to decompose the task into smaller steps, even in novel domains not explicitly present in their training data. Reasoning models also excel at the long-sequence tasks, but no single model attains perfect accuracy on all tasks, and most models display near-random performance on relatively short image sequences. This indicates that regardless of modality, solving MET-Bench tasks remains an open problem.

The RL experiments demonstrate that reasoning training can substantially improve entity tracking within a single modality, with Gemma 3 4B IT improving from near-random performance to 92.2% accuracy on text-based Shell Game tracking. However, the lack of robust cross-modal transfer suggests that current VLM architectures learn modality-specific representations rather than unified entity tracking capabilities. Achieving robust multimodal entity tracking may require new methods, such as explicit cross-modal alignment objectives or other inductive biases that encourage models to develop modality-agnostic tracking mechanisms.

## 6.6 Conclusions

Our findings suggest the primary bottleneck in multimodal entity tracking is not visual recognition but sequential reasoning over visual updates. Unlike text-based representations, which align with the models' training paradigms, visual updates require implicit state reconstruction, a task most current architectures do not perform reliably. No model achieves perfect accuracy in all long-sequence experiments, highlighting the need for improved training of frontier models for entity state tracking. RL training of entity state reasoning is a

promising path forward that can improve the performance of open-source models. The lack of robust cross-modal transfer underscores that visual entity tracking requires fundamentally different representations or training objectives.

# Chapter 7: Future Work

This dissertation improves and evaluates the action reasoning abilities of large multimodal models from complementary perspectives. On the improvement side, Chapter 3 enforces executability through external symbolic planning knowledge, Chapter 4 improves sample efficiency through compositional policy representations, and Chapter 6 improves action-effect reasoning through reinforcement learning training. On the evaluation side, Chapter 5 reveals that frontier models cannot reliably reason about action dependencies in text, and Chapter 6 reveals that models struggle to understand the effects of actions on entity states when these are conveyed through visual observations. Each contribution exposes specific gaps that define both immediate next steps and longer-term research directions. The improvement methods of Chapter 3 and Chapter 4 rely on hand-specified planning domains and fixed pretraining curricula; relaxing these assumptions requires methods that acquire structure automatically. The evaluation benchmarks are restricted to cooking recipes and game environments; extending them to richer domains would bolster the generality of the findings. The gap between text-based and image-based entity tracking in Chapter 6 points toward new training objectives and architectures for multimodal reasoning. We organize these directions below into short-term extensions that follow directly from the current work and long-term themes.

## 7.1 Short-term Future Directions

### 7.1.1 Automatic Induction of Planning Domains

The planning-augmented method in Chapter 3 requires a hand-specified planning domain that formally defines action preconditions and effects. Constructing such a specifica-

tion demands programming effort and domain expertise, limiting the method's applicability to environments where this investment is justified, for example when executability or safety guarantees are required. However, in the course of developing the method, we observed that large language models can generate valid planning domain definitions in structured formats including the Planning Domain Definition Language (Fox and Long, 2003). Other work has similar findings, and has used this capability to solve planning problems in planning domains generated by language models (Liu et al., 2023a; Smirnov et al., 2024). Together these results suggest a path forward for *automatic planning domain induction*: using pretrained models to generate domain definitions that are then used by planners. While this technique has been applied to classical, textual planning problems, its application to increasingly complex domains and tasks remains an open problem.

### 7.1.2 Extending Executability Enforcement to New Domains

The evaluation of planning-augmented parsing in this dissertation is limited to recipe semantic parsing. A natural next step is to evaluate whether the executability improvements generalize to instructional domains with richer action spaces and longer-horizon dependencies, such as VirtualHome (Puig et al., 2018) and ALFRED (Shridhar et al., 2020). Extensions to household task completion, web navigation, and software-control instructions would test the generality of the approach and establish whether symbolic planning knowledge provides consistent benefits across diverse sequential decision-making settings.

### 7.1.3 Eliminating the Pretraining Curriculum for Compositional Policies

The CERLLA method in Chapter 4 requires a pretraining phase in which a hand-specified task decomposition and curriculum is used to learn the basis set of world value functions (WVFs). One important extension is to reduce this manual specification and learn

the WVFs and the language-instruction semantic parser jointly from environment rollouts on randomly sampled tasks, without a fixed curriculum. This is a challenging joint optimization problem: the WVFs must form a basis that covers the space of language-specified tasks, while the semantic parser must learn to compose them correctly given only task-success feedback. Novel value-function approximation and optimization approaches may be required to make the automated learning of this basis tractable.

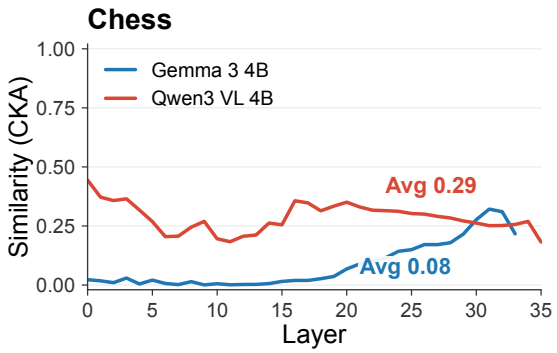
### 7.1.4 Broadening Dependency Evaluation Across Domains and Models

The CaT-Bench evaluation in Chapter 5 considers only cooking recipes as procedural texts. The methods and question templates are domain-general, however, and can be applied to medical practice guidelines, repair manuals, software tutorials, and other procedural domains. Extending the benchmark to these settings would test whether the dependency-reasoning failures we observe are a more generalizable limitation of frontier models.

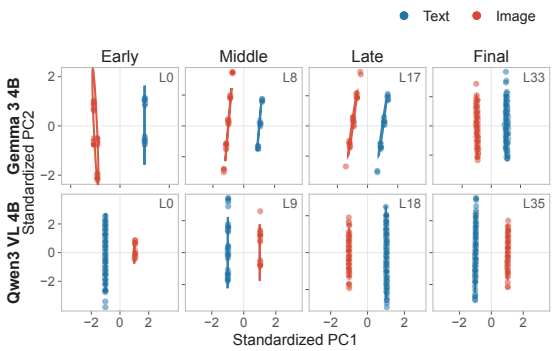
### 7.1.5 Interpretability Analysis of Multimodal Entity Tracking Deficits

Chapter 6 reveals a substantial gap between text-based and image-based entity tracking, even when models can accurately perceive individual visual actions. The source of this deficit within the model remains unclear. To compare the model's text and image representations, we collect a pooled hidden-state vector for each valid Shell Game instance from the corresponding text input and image input at each transformer layer. Centered kernel alignment (CKA) compares the similarity structure of activation spaces (Kornblith et al., 2019), while principal component analysis (PCA) projects hidden states onto directions of maximal variance (Jolliffe and Cadima, 2016), providing a two-dimensional view of the image-text modality gap (Liang et al., 2022). As shown in Figure 7.1, these analyses characterize how text and image representations differ across model layers. While the

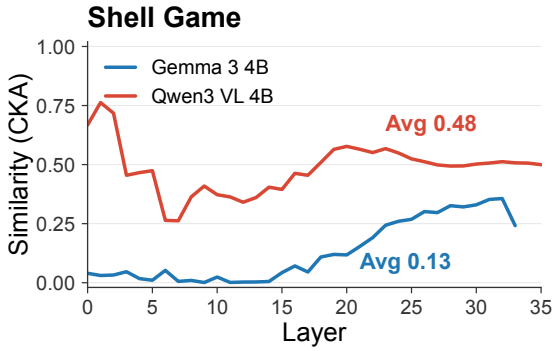
representations differ and this may be responsible for the failure of the models to integrate text and image entity state updates, it is not clear why the representations differ, and what these differences mean for task performance. Future work should apply targeted probing and mechanistic interpretability methods (Lin et al., 2025) to identify whether the deficit originates in visual encoding, cross-modal state integration, or sequential reasoning over visual inputs. Localizing the failure mode would inform the design of architectural modifications or training objectives that specifically address it.



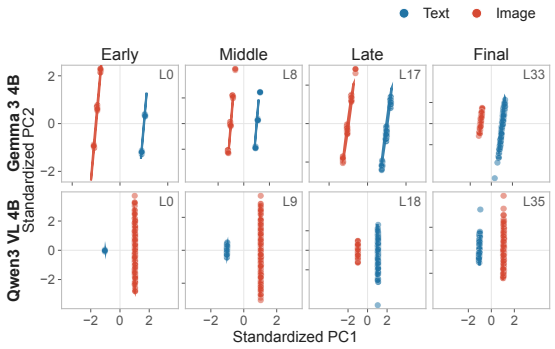

(a) Layer-wise CKA similarity for Chess.




(b) PCA projection across layers for Chess.




(c) Layer-wise CKA similarity for Shell Game.




(d) PCA projection across layers for Shell Game.

Figure 7.1: Representation analyses for Chess and Shell Game entity tracking in MET-Bench. The plots compare text and image representations across transformer layers. While the inputs are semantically equivalent and require the same entity-tracking response, modality-specific structure remains visible.

## 7.2 Long-term Future Directions

### 7.2.1 Compositional Policy Learning Beyond Grid Worlds

The CERLLA method demonstrates compositional generalization in a discretized MiniGrid-BabyAI environment with a small action space and simple visual observations. A critical long-term question is whether compositional value functions and language-guided recombination transfer to sequential decision-making tasks with continuous action spaces, rich visual observations, and real-world physical constraints. Vision-and-language navigation (Anderson et al., 2018) and robotic manipulation (Zeng et al., 2020) are naturally compositional in their goal specifications and have been studied in discretized forms. Validating compositional policy representations in these settings would establish whether the sample-efficiency and generalization benefits observed in Chapter 4 extend to the tasks where they are most needed.

### 7.2.2 Multimodal Entity Tracking for Embodied Decision Making

The ability to track entity states across modalities is a prerequisite for effective embodied agents: a robot must integrate visual observations of its workspace with linguistic goal specifications to maintain a coherent world model and predict the consequences of its actions. The lack of robust cross-modal transfer observed in Chapter 6 suggests that current vision-language architectures do not develop unified entity-tracking representations during standard pretraining. Achieving robust multimodal entity tracking may require new training objectives, such as explicit cross-modal alignment losses, auxiliary state-prediction tasks, or memory mechanisms that maintain entity representations across modalities and time steps. Empirical studies evaluating whether improved entity state tracking leads to higher task success rates, improved sample efficiency, or stronger generalization in embodied environments would establish the practical impact of progress on this problem.

### 7.2.3 Unified Action Reasoning for Deployed Agents

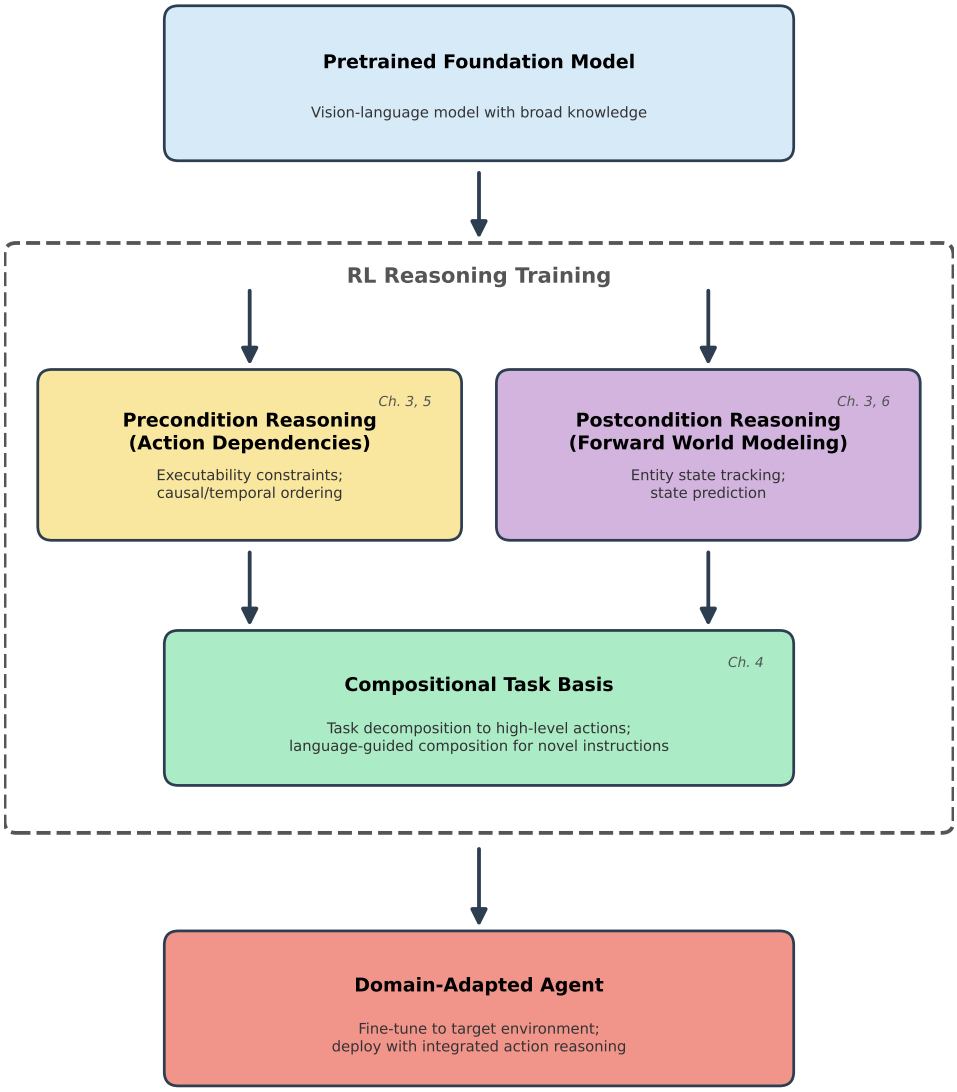



Figure 7.2: Proposed training pipeline for a unified action reasoning agent. To build an agent with robust action reasoning, a pretrained foundation model undergoes RL reasoning training along two complementary axes: precondition reasoning, which teaches executability constraints and causal/temporal dependencies (Chapters 3 and 5), and postcondition reasoning, which develops entity state tracking and forward world modeling (Chapters 3 and 6). These capabilities feed into compositional task basis construction (Chapter 4), where the agent learns a task set of composable high-level actions. The resulting system is then adapted to a target deployment domain.

Each chapter of this dissertation addresses a different facet of action reasoning in isolation: executability, compositional generalization, dependency understanding, and multimodal state tracking. A long-term goal is to unify these capabilities within a single agent architecture that can verify that its plans respect causal and temporal dependencies, maintain coherent entity representations across text and visual observations, and compose reusable skills to solve novel instructions. Figure 7.2 illustrates how these capabilities, each developed in a separate chapter, could be integrated within a single agent that receives language instructions and visual observations and produces actions in an environment. Achieving this integration requires auxiliary training objectives that encourage models to maintain consistent entity state representations across modalities, understand and verify the precondition relationships between actions, and leverage compositional value functions to improve sample efficiency. Progress toward such unified systems would represent a significant advance in building decision-making agents with robust action reasoning abilities.

# Chapter 8: Conclusion

As large multimodal models are increasingly deployed as components of decision-making agents, the ability to reason about actions becomes a critical bottleneck. Agents must understand how actions change environment states, what constraints govern their valid execution, how actions compose to achieve goals, and how to track the effects of actions across modalities. In this dissertation, we develop methods that improve action reasoning by integrating external knowledge and compositional structure into pretrained models, and benchmarks that rigorously evaluate action reasoning abilities, revealing both strengths and significant limitations of frontier models.

Our methods demonstrate that integrating external symbolic knowledge and compositional structure into pretrained models can substantially improve performance for sequential decision-making tasks. Our benchmarks reveal that frontier models have significant limitations in reasoning about action dependencies and tracking entity states across modalities, pointing to important directions for future work. We believe this dissertation provides a foundation for building decision-making agents with robust action reasoning abilities, and will motivate future research in this direction.

# Works Cited


Michael Ahn, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

Anthropic. System Card: Claude Opus 4.5, 2025. URL `https://www.anthropic.com/claude-opus-4-5-system-card`.

Anthropic. Claude 3.7 Sonnet. Anthropic Announcement (Feb. 24, 2025), 2025. URL `https://www.anthropic.com/news/claude-3-7-sonnet`.

Anthropic. System Card: Claude Sonnet 4.6, 2026. URL `https://www.anthropic.com/claude-sonnet-4-6-system-card`.

Mohaddeseh Bastan, Mahnaz Koupaee, Youngseo Son, Richard Sicoli, and Niranjan Balasubramanian. Author's sentiment prediction. In Donia Scott, Nuria Bel,

and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 604–615, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.52. URL https://aclanthology.org/2020.coling-main.52/.

William E Bishop, Alice Li, Christopher Rawles, and Oriana Riva. Latent state estimation helps ui agents to reason, 2024. URL https://arxiv.org/abs/2405.11120.

Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A Knepper, and Yoav Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on Robot Learning*, pages 1415–1438. PMLR, 2020.

Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

Florian Bordes, Quentin Garrido, Justine T Kao, Adina Williams, Michael Rabbat, and Emmanuel Dupoux. IntPhys 2: Benchmarking intuitive physics understanding in complex synthetic environments, 2025. URL https://arxiv.org/abs/2506.09849.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. *ICLR*, 2018.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D. Hwang, Xiang Lorraine Li, Hirona J. Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. Plasma: Making small language models better procedural knowledge models for (counterfactual) planning, 2023.

Satchuthananthavale RK Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, 2009.

SRK Branavan, Luke Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1268–1277, 2010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

Razvan Bunescu and Marius Paşca. Using encyclopedic knowledge for named entity disambiguation. In Diana McCarthy and Shuly Wintner, editors, *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, Trento, Italy, April 2006. Association for Computational Linguistics. URL https://aclanthology.org/E06-1002.

Ozan Caglayan, Pranava Madhyastha, and Lucia Specia. Curious case of language generation evaluation metrics: A cautionary tale. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2322–2328, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.210. URL https://aclanthology.org/2020.coling-main.210/.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/4c7a167bb329bd92580a99ce422d6fa6-Paper.pdf.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph

Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

Thomas Carta, Pierre-Yves Oudeyer, Olivier Sigaud, and Sylvain Lamprier. Eager: Asking and answering questions for automatic reward shaping in language-guided rl. *Advances in Neural Information Processing Systems*, 35:12478–12490, 2022.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32:1, 2018.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Evaluating question answering evaluation. In Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen, editors, *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5817. URL https://aclanthology.org/D19-5817/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019.

Vanya Cohen and Raymond Mooney. Using planning to improve semantic parsing of instructional texts. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei, editors, *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 47–58, Toronto, Canada, June 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlrse-1.5. URL https://aclanthology.org/2023.nlrse-1.5.

Vanya Cohen and Raymond Mooney. Met-bench: Multimodal entity tracking for evaluating the limitations of vision-language and reasoning models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML 2026)*, 2026. URL https://arxiv.org/abs/2502.10886.

Vanya Cohen, Jason Xinyu Liu, Raymond Mooney, Stefanie Tellex, and David Watkins. A survey of robotic language grounding: Tradeoffs between symbols and embeddings, 2024.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana, June 2018. Association for Computational

Linguistics. doi: 10.18653/v1/N18-1144. URL https://aclanthology.org/N18-1144/.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. Everything happens for a reason: Discovering the purpose of actions in procedural text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4496–4505, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1457. URL https://aclanthology.org/D19-1457.

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. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, 2019.

Aissatou Diallo, Antonis Bikakis, Luke Dickens, Anthony Hunter, and Rob Miller. Pizzacommonsense: Learning to model commonsense reasoning about intermediate

steps in cooking recipes, 2024.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, Fengli Xu, and Yong Li. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, pages 1–38, 2026. ISSN 0360-0300. doi: 10.1145/3746449. URL https://doi.org/10.1145/3746449.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *2009 IEEE International Conference on Robotics and Automation*, pages 4163–4168. IEEE, 2009.

Erwan Fagnou, Paul Caillon, Blaise Delattre, and Alexandre Allauzen. Chain and causal attention for efficient entity tracking. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13174–13188, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.731. URL https://aclanthology.org/2024.emnlp-main.731/.

Xiaokun Feng, Shiyu Hu, Kaiqi Huang, Xuchen Li, Meiqi Wu, Dailing Zhang, and Xin Zhao. A multi-modal global instance tracking benchmark (mgit): Better locating

target in complex spatio-temporal and causal relationship. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 25007–25030. Curran Associates, Inc., 2023. doi: 10.52202/075280-1087.

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

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

Leo Gao. On the sizes of openai api models. https://blog.eleuther.ai/gpt3-model-sizes/, 2021. Accessed: 2022-08-11.

Qiyue Gao, Xinyu Pi, Kevin Liu, Junrong Chen, Ruolan Yang, Xinqi Huang, Xinyu Fang, Lu Sun, Gautham Kishore, Bo Ai, Stone Tao, Mengyang Liu, Jiaxi Yang, Chao-Jung Lai, Chuanyang Jin, Jiannan Xiang, Benhao Huang, Zeming Chen, David Danks, Hao Su, Tianmin Shu, Ziqiao Ma, Lianhui Qin, and Zhiting Hu. Do vision-language models have internal world models? towards an atomic evaluation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26170–26195, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1342. URL https://aclanthology.org/2025.findings-acl.1342/.

Gemini Team. Gemini 2.5 Pro, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Pro-Model-Card.pdf.

GemmaTeam, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd,

Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL `https://arxiv.org/abs/2503.19786`.

Michael P Georgeff. *Reasoning About Actions & Plans*. Elsevier, 1987.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

Matthew L Ginsberg and David E Smith. Reasoning about action i: A possible worlds approach. *Artificial intelligence*, 35(2):165–195, 1988.

Nakul Gopalan, Dilip Arumugam, Lawson Wong, and Stefanie Tellex. Sequence-to-sequence language grounding of non-markovian task specifications. *Robotics: Science and Systems XIV*, 2018.

Aditya Gupta and Greg Durrett. Effective use of transformer networks for entity tracking. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 759–769, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1070. URL https://aclanthology.org/D19-1070/.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation*, pages 6244–6251. IEEE, 2018.

Xuehai He, Weixi Feng, Kaizhi Zheng, Yujie Lu, Wanrong Zhu, Jiachen Li, Yue Fan, Jianfeng Wang, Linjie Li, Zhengyuan Yang, Kevin Lin, William Yang Wang, Lijuan Wang, and Xin Eric Wang. Mmworld: Towards multi-discipline multi-faceted world model evaluation in videos. In *Proceedings of the 2025 International Conference on Learning Representations*, 2025.

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26(1):191–246, July 2006. ISSN 1076-9757.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. *ICLR*, 2017.

Jerry R. Hobbs. Resolving pronoun references. *Lingua*, 44(4):311–338, 1978. ISSN 0024-3841. doi: https://doi.org/10.1016/0024-3841(78)90006-2. URL https://www.sciencedirect.com/science/article/pii/0024384178900062.

J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, May 2001. ISSN 1076-9757. doi: 10.1613/jair.855. URL http://dx.doi.org/10.1613/jair.855.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Zhaoyi Joey Hou, Li Zhang, and Chris Callison-Burch. Choice-75: A dataset on decision branching in script learning. *arXiv preprint arXiv:2309.11737*, 2023.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In Brian Davis, Yvette Graham, John Kelleher, and Yaji Sripada, editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.inlg-1.23. URL https://aclanthology.org/2020.inlg-1.23/.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision*, pages 53–69, 2018.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

Jonathan Hunt, Andre Barreto, Timothy Lillicrap, and Nicolas Heess. Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, pages 2911–2920, 2019.

Yuqian Jiang, Shiqi Zhang, Piyush Khandelwal, and Peter Stone. Task planning in robotics: an empirical comparison of pddl-based and asp-based systems, 2019.

Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 04 2016. ISSN 1364-503X. doi: 10.1098/rsta.2015.0202. URL https://doi.org/10.1098/rsta.2015.0202.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1114. URL https://aclanthology.org/D15-1114.

Najoung Kim and Sebastian Schuster. Entity tracking in language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.213. URL https://aclanthology.org/2023.acl-long.213.

Najoung Kim, Sebastian Schuster, and Shubham Toshniwal. Code pretraining improves entity tracking abilities of language models. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2405.21068.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 2019.

Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Compositional RL agents that follow language commands in temporal logic. *Frontiers in Robotics and AI*, 8:689550, 2021.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Ray Mooney. CaT-bench: Benchmarking language model understanding of causal and temporal dependencies in plans. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19336–19354, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1077. URL https://aclanthology.org/2024.emnlp-main.1077/.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. 2024.

Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994. URL https://aclanthology.org/J94-4002.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, USA, 2006. ISBN 0521862051.

Duong Minh Le, Ruohao Guo, Wei Xu, and Alan Ritter. Improved instruction ordering in recipe-grounded conversation, 2023.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.143. URL https://aclanthology.org/2021.acl-long.143.

Belinda Z Li, Zifan Carl Guo, and Jacob Andreas. (how) do language models track state? *International Conference on Machine Learning*, 2025a.

Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal foundation models: From specialists to general-purpose assistants, 2023a. URL https://arxiv.org/abs/2309.10020.

Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph E. Gonzalez, Ion Stoica, Song Han, and Yao Lu. Worldmodelbench: Judging video generation models as world models. In *Advances in Neural Information Processing Systems*, 2025b.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=DeG07_TcZvT.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17612–17625. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/702f4db7543a7432431df588d57bc7c9-Paper-Conference.pdf.

Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

Zihao Lin, Samyadeep Basu, Mohammad Beigi, Varun Manjunatha, Ryan A. Rossi, Zichao Wang, Yufan Zhou, Sriram Balasubramanian, Arman Zarei, Keivan Rezaei, Ying Shen, Barry Menglong Yao, Zhiyang Xu, Qin Liu, Yuxiang Zhang, Yan Sun, Shilong Liu, Li Shen, Hongxuan Li, Soheil Feizi, and Lifu Huang. A survey on mechanistic interpretability for multi-modal foundation models, 2025. URL https://arxiv.org/abs/2502.17516.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023a. URL https://arxiv.org/abs/2304.11477.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

Ruicheng Liu, Rui Mao, Anh Tuan Luu, and Erik Cambria. A brief survey on recent advances in coreference resolution. *Artificial Intelligence Review*, 56(12): 14439–14481, 2023b.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*. PMLR, 2025. URL https://arxiv.org/abs/2410.21333. arXiv preprint arXiv:2410.21333.

Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1138. URL https://aclanthology.org/P16-1138.

Qingsong Ma, Johnny Wei, OndVrej Bojar, and Yvette Graham. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume*

*2: Shared Task Papers, Day 1)*, pages 580–590. Association for Computational Linguistics, 2019.

Donata Marasini, Piero Quatto, and Enrico Ripamonti. Assessing the inter-rater agreement for ordinal data through weighted indexes. *Statistical Methods in Medical Research*, 25:2611–2633, 2016. doi: 10.1177/0962280214529560.

John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationary Office. URL http://www-formal.stanford.edu/jmc/mcc59.html.

Jorge A. Mendez, Marcel Hussing, Meghna Gummadi, and Eric Eaton. CompoSuite: A compositional reinforcement learning benchmark. In *1st Conference on Lifelong Learning Agents (CoLLAs-22)*, 2022.

Jorge Mendez-Mendez and Eric Eaton. How to reuse and compose knowledge for a lifetime of tasks: A survey on continual learning and functional composition. *Transactions on Machine Learning Research (TMLR)*, 2023.

Meta. Llama 4 Model Card. Meta Model Card (April 5, 2025), 2025. URL https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, Jin Zhu, Jingtao Han, Jingyang Li, Junbin Xie, Junhao Xu, Junjie Yan, Kaishun Zhang, Kecheng Xiao, Kexi Kang, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin

Zheng, Linbo Chai, Long Xing, Meizhi Ju, Mingyuan Chi, Mozhi Zhang, Peikai Huang, Pengcheng Niu, Pengfei Li, Pengyu Zhao, Qi Yang, Qidi Xu, Qiexiang Wang, Qin Wang, Qiuhui Li, Ruitao Leng, Shengmin Shi, Shuqi Yu, Sichen Li, Songquan Zhu, Tao Huang, Tianrun Liang, Weigao Sun, Weixuan Sun, Weiyu Cheng, Wenkai Li, Xiangjun Song, Xiao Su, Xiaodong Han, Xinjie Zhang, Xinzhu Hou, Xu Min, Xun Zou, Xuyang Shen, Yan Gong, Yingjie Zhu, Yipeng Zhou, Yiran Zhong, Yongyi Hu, Yuanxiang Fan, Yue Yu, Yufeng Yang, Yuhao Li, Yunan Huang, Yunji Li, Yunpeng Huang, Yunzhi Xu, Yuxin Mao, Zehan Li, Zekang Li, Zewei Tao, Zewen Ying, Zhaoyang Cong, Zhen Qin, Zhenhua Fan, Zhihang Yu, Zhuo Jiang, and Zijia Wu. Minimax-01: Scaling foundation models with lightning attention, 2025. URL https://arxiv.org/abs/2501.08313.

Mistral. Mistral Small 3.2, 2025. URL https://docs.mistral.ai/models/model-cards/mistral-small-3-2-25-06.

Eric Mitchell, Joseph Noh, Siyan Li, Will Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher Manning. Enhancing self-consistency and performance of pre-trained language models through natural language inference. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1768, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.115. URL https://aclanthology.org/2022.emnlp-main.115/.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*,

518(7540):529–533, 2015.

Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A Boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Geraud Nangue Tasse, Steven James, and Benjamin Rosman. World value functions: Knowledge representation for multitask reinforcement learning. In *The 5th Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2022.

Nguyen Nguyen, Jing Bi, Ali Vosoughi, Yapeng Tian, Pooyan Fazli, and Chenliang Xu. OSCaR: Object state captioning and state change representation. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3565–3576, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.226. URL https://aclanthology.org/2024.findings-naacl.226/.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

OpenAI. Gpt-4 technical report, 2023.

OpenAI. Introducing GPT-4.1 in the API. OpenAI Technical Report (April 14, 2025), 2025a. URL https://openai.com/index/gpt-4-1/.

OpenAI. Introducing OpenAI o3 and o4-mini. OpenAI Release Notes (April 16, 2025), 2025b. URL https://openai.com/index/introducing-o3-and-o4-mini/.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia

Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy,

Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas

Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. GPT-4o System Card, 2024a. URL https://arxiv.org/abs/2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song,

Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan,

Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. OpenAI o1 System Card, 2024b. URL https://arxiv.org/abs/2412.16720.

Liang-Ming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yugang Jiang, and Tat-Seng Chua. Multi-modal cooking workflow construction for food recipes. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1132–1141, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379885. doi: 10.1145/3394171.3413765. URL https://doi.org/10.1145/3394171.3413765.

Dim P. Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, and Antonio Torralba. Learning program representations for food images and cooking recipes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16559–16569, June 2022.

Pinelopi Papalampidi, Kris Cao, and Tomas Kocisky. Towards coherent and consistent use of entities in narrative generation. In *International Conference on Machine Learning*, pages 17278–17294. PMLR, 2022.

Paolo Pareti, Benoit Testu, Ryutaro Ichise, Ewan Klein, and Adam Barker. Integrating know-how into the linked data cloud. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 385–396. Springer, 2014.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2402.14811.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

Qwen Team. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

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

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009. URL https://api.semanticscholar.org/CorpusID:207178704.

Pascal J Sager, Benjamin Meyer, Peng Yan, Rebekka von Wartburg-Kottler, Layan Etaiwi, Aref Enayati, Gabriel Nobel, Ahmed Abdulkadir, Benjamin F Grewe, and Thilo Stadelmann. A comprehensive survey of agents for computer use: Foundations, challenges, and future directions. *arXiv preprint arXiv:2501.16150*, 2025.

Shailaja Keyur Sampat, Maitreya Patel, Subhasish Das, Yezhou Yang, and Chitta Baral. Reasoning about actions over visual and linguistic modalities: A survey, 2022. URL https://arxiv.org/abs/2207.07568.

R.C. Schank and R.P. Abelson. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures, 1977a.

Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977b.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, Antonio Torralba, and Stephanie Fu. A vision check-up for language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, pages 14410–14419, 2024. doi: 10.1109/CVPR52733.2024.01366. URL https://openaccess.thecvf.com/content/CVPR2024/html/Sharma_A_Vision_Check-up_for_Language_Models_CVPR_2024_paper.html.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*. PMLR, 2018.

Richard Shin and Benjamin Van Durme. Few-shot semantic parsing with language models trained on code. *arXiv preprint arXiv:2112.08696*, 2021.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Ben Van Durme. Constrained language models yield few-shot semantic parsers. In *2021 Empirical Methods in Natural Language Processing*, November 2021. URL https://www.microsoft.com/en-us/research/publication/constrained-language-models-yield-few-shot-semantic-parsers/.

Keisuke Shirai, Atsushi Hashimoto, Taichi Nishimura, Hirotaka Kameko, Shuhei Kurita, Yoshitaka Ushiku, and Shinsuke Mori. Visual recipe flow: A dataset for learning visual state changes of objects with recipe flows. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th*

*International Conference on Computational Linguistics*, pages 3570–3577, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.315/.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL https://arxiv.org/abs/1912.01734.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. CLIPort: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2021a.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b. URL https://arxiv.org/abs/2010.03768.

Andrew Silva, Nina Moorman, William Silva, Zulfiqar Zaidi, Nakul Gopalan, and Matthew Gombolay. Lancon-learn: Learning with language to enable generalization in multi-task manipulation. *IEEE Robotics and Automation Letters*, 7(2):1635–1642, 2021.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi

Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi

Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Y. Guan, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll,

Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomek Korbak, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad

Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. OpenAI gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

John Slaney and Sylvie Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125 (1):119–153, 2001. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(00)00079-5. URL https://www.sciencedirect.com/science/article/pii/S0004370200000795.

Pavel Smirnov, Frank Joublin, Antonello Ceravola, and Michael Gienger. Generating consistent pddl domains with large language models, 2024. URL https://arxiv.org/abs/2404.07751.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112

(1–2):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Zoltán Gendler Szabó. Compositionality. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2020 edition, 2020.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. A dataset for tracking entities in open domain procedural text. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.520. URL https://aclanthology.org/2020.emnlp-main.520/.

Dan Tasse and Noah A Smith. Sour cream: Toward semantic processing of recipes. *Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-LTI-08-005*, 2008.

Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pages 1369–1376, 2007.

Emanuel Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pages 1856–1864, 2009.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11385–11393, 2022.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.

Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pages 6401–6409, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13763–13773, June 2021.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori

Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. The generative AI paradox: "what it can create, it may not understand". In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CF8H8MS5P8.

Edward Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *IEEE International Conference on Robotics and Automation*, pages 4430–4436, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System*

*Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, 2006.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals, 2024.

Zihui Xue, Kumar Ashutosh, and Kristen Grauman. Learning object state changes in videos: An open-world perspective. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Yoko Yamakata, Shinsuke Mori, and John Carroll. English recipe flow graph corpus. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.638.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities, 2023. URL https://arxiv.org/abs/2303.04129.

Takuma Yoneda, Jiading Fang, Peng Li, Huanyu Zhang, Tianchong Jiang, Shengjie Lin, Ben Picker, David Yunis, Hongyuan Mei, and Matthew R Walter. Statler: State-

maintaining language models for embodied reasoning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15083–15091. IEEE, 2024.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. Reinforcement world model learning for llm-based agents, 2026. URL https://arxiv.org/abs/2602.05842.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2020.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. Causal reasoning of entities and events in procedural texts. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.31. URL https://aclanthology.org/2023.findings-eacl.31.

Tianyi Zhang, Li Zhang, Zhaoyi Hou, Ziyu Wang, Yuling Gu, Peter Clark, Chris Callison-Burch, and Niket Tandon. Proc2pddl: Open-domain planning representations from texts, 2024.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. Natural plan: Benchmarking llms on natural language planning, 2024.