Language-Informed Synthesis of Rational Agent Models for Grounded Theory-of-Mind Reasoning On-The-Fly

Lance Ying[1,2], **Ryan Truong**[2], **Katherine M. Collins**[3], **Cedegao E. Zhang**[1], **Megan Wei**[4], **Tyler Brooke-Wilson**[5], **Tan Zhi-Xuan**[1‡], **Lionel Wong**[6‡], **Joshua B. Tenenbaum**[1‡]

[1]MIT [2]Harvard University [3]University of Cambridge
[4]Brown University [5]Yale University [6]Stanford University
‡ co-senior authors

Abstract

Drawing real world social inferences usually requires taking into account information from multiple modalities. Language is a particularly powerful source of information in social settings, especially in novel situations where language can provide both abstract information about the environment dynamics and concrete specifics about an agent that cannot be easily visually observed. In this paper, we propose Language-Informed Rational Agent Synthesis (LIRAS), a framework for drawing context-specific social inferences that integrate linguistic and visual inputs. LIRAS frames multimodal social reasoning as a process of constructing structured but situation-specific agent and environment representations – leveraging multimodal language models to parse language and visual inputs into unified symbolic representations, over which a Bayesian inverse planning engine can be run to produce granular probabilistic judgments. On a range of existing and new social reasoning tasks derived from cognitive science experiments, we find that our model (instantiated with a comparatively lightweight VLM) outperforms ablations and state-of-the-art models in capturing human judgments across all domains.

1 Introduction

Making sense of any real social situation requires integrating many different sources of information. *Language*, in particular, can fundamentally recast our understanding of the social environment around us. Being told the abstract dynamics underlying a social institution, from the rules of American football to the norms of drive-through restaurants, gives us a useful overarching picture of people's goals and intentions in unfamiliar settings. Other times, language can provide specifics about particular people and environments. Hearing that a friend tends to get hungry around midnight and keeps a spare stash of chocolate in the highest pantry shelf, for instance, gives new meaning to a few glimpses of someone bumbling around the kitchen in the dark. These tidbits of socially-relevant information from language allow us to draw much richer, more flexible, and often quite situation-dependent conclusions about the behavior we see.

How do we integrate language with perceptual information to support this kind of *grounded* but often *highly ad-hoc* social reasoning, in which language can flexibly restructure our inferences about the agents we observe? This setting poses particularly acute challenges for two dominant flavors of computational work in social reasoning. For approaches that cast social reasoning as principled inferences over structured models of agents and environments, as in many symbolic AI and cognitive science frameworks (e.g., Baker et al. 2011; Jara-Ettinger et al. 2016), this setting tests the scalability and breadth that can be achieved using (usually hand-engineered) symbolic models of particular environments and domains. For approaches that use large-scale neural models trained on language and visual inputs, like recent LLM and VLM-based systems (e.g., Kosinski 2024; Jin et al. 2024), this setting tests the generalizability of complex decision-making and latent inference over particularly novel inputs. Ongoing evaluations suggest that *each* component of grounded, ad-hoc social reasoning poses challenges for both dominant approaches (Hu et al., 2025; Schulze Buschoff et al., 2025; Jin et al., 2024) – challenges that only compound in the harder multi-modal setting.

In this paper, we develop **Language-Informed Rational Agent Synthesis (LIRAS)**, a framework that can **integrate linguistic and visual inputs** to draw **ad-hoc, probabilistic inferences** about agents' mental states in grounded settings. Our approach achieves this by using neural models to parse language and visual inputs into unified symbolic representations, which support automatic Bayesian inference and inverse planning. Impor-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 12217–12235
November 4-9, 2025

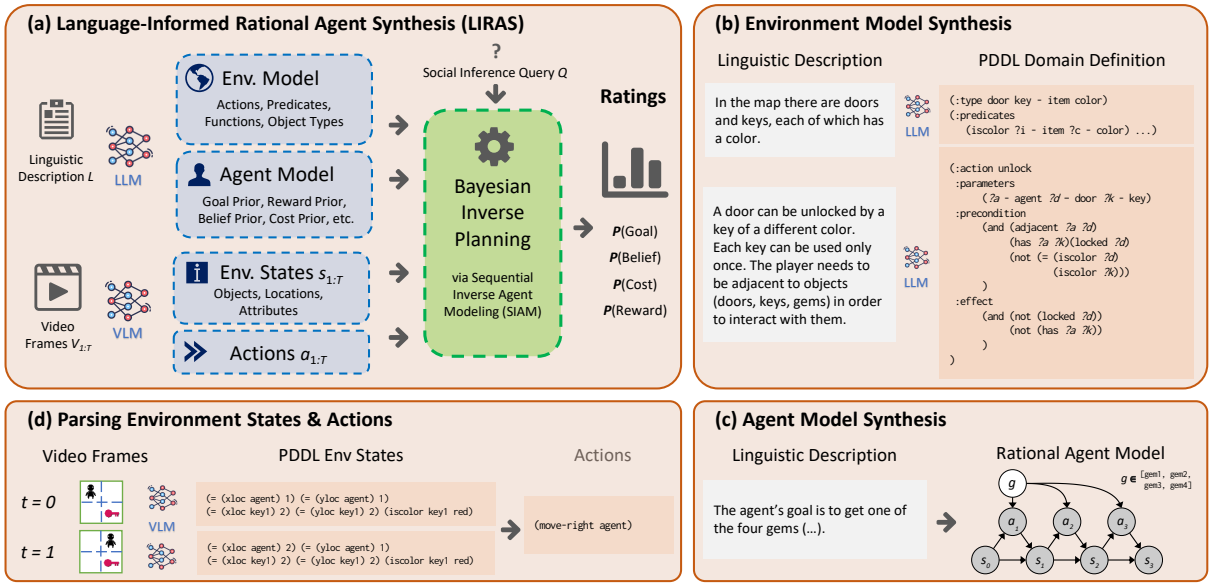

Figure 1: Overview of Language-Informed Rational Agent Synthesis (LIRAS). **(a)** Overall architecture. **(b)** LIRAS uses a vision-capable large language model (LLM) to parse linguistic descriptions into a domain-specific environment model expressed in PDDL, which specifies object types, predicates, functions and action definitions. **(c)** LIRAS constructs a rational agent model based on the linguistic descriptions about the agent, including the space of the agent's goals, beliefs, etc. **(d)** LIRAS parses visual inputs into symbolic PDDL environment states, and derives agent actions from those states.

tantly, our approach casts multimodal social reasoning as a process of constructing small but structured environment representations *on-the-fly*, tailored to a particular set of inputs. Rather than hand-construct domain-specific world models, or seek to derive universal features for visual understanding, our approach lets new, relevant details from language change how we parse and reason about any particular visual scene.

We evaluate our approach on a suite of popular domains used for social cognition experiments, as well as novel variants designed to probe people and models' generalization capacities under new environment dynamics and abstract rule structures. We demonstrate that our framework, even when built with a lightweight VLM, can capture human-like social reasoning in a context-sensitive way across each of these domains. In contrast, we find that much larger state-of-the-art VLMs, such as OpenAI's o3 model, often fail to reliably integrate linguistic details with visual observations and generally are much more uneven in capturing human judgments across these domains.

2 Language-Informed Rational Agent Synthesis (LIRAS)

In this paper, we consider how to solve ad-hoc social reasoning tasks that are grounded in visual observations of an agent taking actions over time. Each task $(L, Q, V_{1:T})$ is defined by a linguistic description L of the agent and its environment, a social inference query Q expressed in natural language, and a sequence of $T+1$ video frames $V_{0:T}$ showing how the agent interacts with the environment over time. Given $(L, Q, V_{0:T})$, LIRAS produces $k \geq 1$ graded ratings $R_{1:k} \in \mathbb{R}^k$ about the agent's mental states (e.g. goals, beliefs, etc.) based on the query Q. Since probabilistic social reasoning lacks a "ground-truth" answer (Baker et al., 2017; Ying et al., 2025a), we assess task performance based on how *human-like* the ratings $R_{1:k}$ are, measuring correlation with a dataset of collected human responses (see Section 3).

To solve these tasks, we propose *Language-Informed Rational Agent Synthesis* (LIRAS, Figure 1). LIRAS differs from social reasoning methods which rely on either prompting a single LLM (Sap et al., 2022; Moghaddam and Honey, 2023) or scaffolding of LLM calls within a structured probabilistic inference procedure (Cross et al., 2024; Kim et al., 2025; Zhang et al., 2025). Instead, given a language description L, LIRAS synthesizes a *rational model of an agent and its environment* — which assigns likelihoods to an agents' actions by solving the (Partially Observable) Markov Decision Process ((PO)MDP) (Bellman, 1958; Kaelbling et al.,

1998) that describes the agent's decision problem — and draws probabilistic inferences about mental states from the agent's actions via *Bayesian inverse planning* with respect to that agent model (Baker et al., 2009, 2017; Zhi-Xuan et al., 2020). This process is decomposed into four components: (i) *synthesis of an environment model* (Fig. 1b); (ii) *synthesis of a rational agent model* (Fig. 1c); (iii) *parsing environment states and agent actions* (Fig. 1d); (iv) *mental state inference via Sequential Inverse Agent Modeling (SIAM)* (Fig. 1a, right), a flexible engine for Bayesian inverse planning that extends Sequential Inverse Plan Search (Zhi-Xuan et al., 2020). We describe each component below.

2.1 Synthesizing Environment Models

In order to synthesize a rational agent model, we first need to synthesize a *environment model* or *world model* that the agent model is situated within. Following work that translates language into symbolic world models represented as programs (Wong et al., 2023; Tang et al., 2024; Wong et al., 2024; Liu et al., 2023; Xie et al., 2023), LIRAS achieves this by using an LLM to translate the task description L into a *planning domain* $\mathcal{D}$ represented in the Planning Domain Definition Language (PDDL) (Aeronautiques et al., 1998; Zhi-Xuan, 2022). Given L and an instruction prompt I_{env}, we rejection sample from the LLM to ensure syntactic and semantic validity of the generated PDDL domain:

$$\mathcal{D} \sim P_{\text{LLM}}(\mathcal{D}|L, I_{\text{env}}, \mathcal{D} \text{ is valid}) \quad (1)$$

As shown in Figure 1(b), a planning domain $\mathcal{D} = (\mathcal{T}, \mathcal{P}, \mathcal{A})$ consists of a set of object types $\mathcal{T}$, predicates $\mathcal{P}$, and action templates $\mathcal{A}$. Given a specific set of (typed) objects $\mathcal{O}$, a planning domain defines a concrete environment $\mathcal{E} = (\mathsf{S}, \mathsf{A}, P_s)$, where S is the set of possible environment states formed from predicates defined over objects in $\mathcal{O}$, A is the set of possible actions derived by filling in action templates with object arguments and $P_s(s_t|s_{t-1}, a_t)$ is an environment transition distribution. This gives us the basic structure on top of which we can define an agent model and perform visual parsing of environment states.

2.2 Synthesizing Rational Agent Models

When people reason about the mental states of another agent, we form an implicit model of that agent that predicts and explains their actions in light of their goals, beliefs, and other mental states (Dennett, 1981). Importantly, we assume that the agent is approximately *rational* — their *beliefs are consistent* with what they observe, and they take *efficient actions* to achieve their goals and satisfy their desires. To formalize this, we follow work in Bayesian theory-of-mind (Baker et al., 2017; Jara-Ettinger et al., 2016; Alanqary et al., 2021; Ying et al., 2025b), treating an agent as a generative processes of the following form:

$$\textit{Mental Prior: } m_0 \sim P_{\theta_0}(m_0; s_0) \quad (2)$$

$$\textit{Mental Update: } m_t \sim P_{\theta_m}(m_t|s_{t-1}, m_{t-1}) \quad (3)$$

$$\textit{Action Selection: } a_t \sim P_a(a_t|m_t, s_{t-1}) \quad (4)$$

Here, m_t represents the mental state of the agent at step t, which can include beliefs b_t, goals g_t, rewards r_t that the agent assigns to achieving a goal, or the perceived costs c_t of certain actions. At each step t, the agent may update their beliefs $b_t \in m_t$ based on their observations in s_{t-1} in a way that preserves *consistency*: If some predicate p is observed to hold true in state s_{t-1}, then p must also hold true in the updated belief b_t. They then take an action a_t to *efficiently* achieve the goals or rewards specified in m_t while minimizing costs. Specifically, we assume that a_t follows a Boltzmann-rational distribution:

$$P_a(a_t|m_t, s_{t-1}) \propto \exp\left(\hat{Q}_{m_t}(s_{t-1}, a_t)\right) \quad (5)$$

where $\hat{Q}_m(s_{t-1}, a_t)$ is an estimate of the *expected utility* of reaching any goal in mental state m_t by taking action a_t at state s. Action a_t then causes a change in the environment $s_t \sim P_s(s_t|s_{t-1}, a_t)$ per the environment model described earlier, and the process repeats.

Depending on the situation, an observer may not need to model all of the agent's mental states. For instance, if the agent has full observability of the environment s_t, there is no need to represent the agents' belief state b_t, since b_t will always agree with s_t. As such, LIRAS constructs agent models in an *ad-hoc* manner: Given the language description L, (and an instruction prompt I_{agent}), we use an LLM to synthesize the parameters $\Theta = (\theta_0, \theta_m)$ that define the agent model (Figure 1(c)):

$$\Theta \sim P_{\text{LLM}}(\Theta|L, I_{\text{agent}}) \quad (6)$$

θ_0 includes information like the space of possible initial beliefs, goals, goal rewards, or action costs (over which we assume a uniform prior), and θ_m includes information like the observability of various objects in state s_t (since this affects how the agent updates their beliefs). By flexibly synthesiz-

ing different models based on the social situation described in L, LIRAS captures human-like flexibility in social reasoning, while preserving the rationality assumptions of belief consistency and action efficiency described above.

2.3 Parsing Environment States and Actions

To infer an agent's mental states, LIRAS first needs to parse the images $I_{0:T}$ into a sequence of symbolically represented environment states $s_{0:T}$ and actions $a_{1:T}$ (Figure 1(d)). We achieve this using a vision-language model (VLM) in a decoding procedure that exploits the grid-based nature of our environments.[1] Specifically, for each grid cell in a video frame V_t, we use the VLM to detect the object in that cell (if any), and generate corresponding PDDL code for the object's properties. This allows us to parse a state s_t from frame V_t:

$$s_t \sim P_{\text{VLM}}(s_t|V_t, I_{\text{states}}) \tag{7}$$

where I_{states} in an instruction prompt. Given the full sequence of environment states $s_{1:t}$, we can then greedily reconstruct each action a_t under the environment model $P_s(s_t|s_{t-1}, a_t)$:

$$a_t = \text{argmax}_a \, P_s(s_t|s_{t-1}, a) \tag{8}$$

2.4 Bayesian Mental State Inference via Sequential Inverse Agent Modeling

Having synthesized an environment model, agent model, environment states $s_{1:T}$ and actions $a_{1:T}$, LIRAS answers the social inference queries in Q via *Bayesian inverse planning*. Specifically, we use Sequential Inverse Agent Modeling (SIAM), an extension of the SIPS probabilistic programming architecture for inverse planning (Zhi-Xuan et al., 2020, 2024a) that supports joint inference over not just goals g (as in Zhi-Xuan et al. (2020)), but also goal preferences/rewards r (similar to Zhi-Xuan et al. (2022)), beliefs (as in belief-space SIPS (Ying et al., 2025b)) and action costs c (similar to Zhi-Xuan et al. (2024b)). SIAM efficiently inverts the rational agent model described in Section 2.2, computing a posterior distribution over the full sequence of latent mental states $m_{0:T}$:

$$P_\Theta(m_{0:T}|s_{0:T}, a_{1:t}) \propto P_{\theta_0}(m_0; s_0)$$
$$\prod_{t=1}^{T} P_{\theta_m}(m_t|s_{t-1}, m_{t-1}) P_a(a_t|m_t, s_{t-1})$$

where the mental state m_t can be a subset of (g_t, b_t, r_t, c_t) depending on the agent model configuration Θ. SIAM achieves efficiency by using incremental planning algorithms to rapidly estimate the expected utility of an action $\hat{Q}_{m_t}(s_{t-1}, a_t)$. We provide details in the Appendix.

Having computed a posterior over all mental states $P_\Theta(m_{0:T}|s_{0:T}, a_{1:t})$, LIRAS can answer a social inference query Q by computing marginal probabilities or posterior expected values. For example, if Q asks for how likely each goal is given the actions, LIRAS returns the marginal posterior $P(g_0|s_{0:T}, a_{1:t})$ over the agent's goal. If Q instead asks for the cost c_0^a of some action $a \in \mathsf{A}$, LIRAS returns the posterior expectation $\mathbb{E}[c_0^a|s_{0:T}, a_{1:t}]$. When Q contains k sub-queries, LIRAS produces k corresponding ratings $R_{1:k}$ from these quantities.

3 Experiments

3.1 Domains

We compare our model and baselines to human social inferences on **existing cognitive science domains** from the social reasoning literature, and a set of **expanded multimodal variants** that are derived from earlier work but that we construct specifically to evaluate the role of language in more complex, grounded environments. The existing experiments have been well-modeled by hand-constructed, domain-specific symbolic models; we choose these to assess how well our approach (and baselines) can capture judgments by synthesizing these structured models from inputs. Collectively, we choose these multi-modal domains to represent a diverse range of social reasoning tasks that vary in features including number of agents, observability, and variables of interest.

The two existing cognitive science domains we consider are:

- **Food trucks** (78 stimuli): a domain from (Baker et al., 2017) in which the instructions describe a student navigating a campus while choosing which (movable) food truck to go to for lunch. The visual stimuli depicts varying paths taken by the agent with partial observability of the trucks. Participants are asked to jointly infer the agents' beliefs and desires.
- **Astronaut** (47 stimuli): a domain from (Jara-Ettinger et al., 2016) in which the instructions describe an astronaut navigating alien terrain to pick up care packages on the way to a space station. The visual stimuli show various colored terrains and paths. Participants must infer

[1] This is not key to our architecture, but simply what we found was necessary given the current unreliability of VLMs and parsing full 2D visual scenes.

the relative costs of walking on each terrain type and the rewards of the care packages.

We also evaluate on an expanded set of multi-modal domains derived from the Doors, Keys, and Gems (DKG) stimuli (introduced in Zhi-Xuan et al. 2020, with a multi-agent version introduced in Ying et al. 2023). The original domains evaluate multi-step planning and inverse planning. Instructions describe an obstacle course in which a player is navigating a maze to reach one of several colored gems, but must first acquire various keys that unlock doors in the maze. We extend both original domains (adding significantly more stimuli to the mutli-agent case), and construct new variants that modify the linguistic instructions which specify the underlying environment dynamics:

- **DKG-Simple** (32 stimuli): One colored key unlocks one door of the same color.
- **DKG-Double** (16 stimuli): Two colored keys unlock one door of the same color.
- **DKG-Reuse** (16 stimuli): One colored key unlocks all doors of the same color.
- **DKG-Inverse** (16 stimuli): One colored key unlocks one door of a different color.
- **Multiagent DKG (m-DKG)** (30 stimuli): Mazes contain two players, a principal and an assistant. The team works together to get one goal gem to the principal agent.

Each of the visual stimuli in the **DKG-Double, DKG-Reuse**, and **DKG-Inverse** variants has an identical corresponding stimulus in the base **DKG-Simple** domain, allowing paired comparison of the role of language in inferences about agent behavior.

3.2 Human Data Collection

For the **food truck** (Baker et al., 2017) and **astronaut** (Jara-Ettinger et al., 2016) domains, we evaluate on the original published data, which only contains *mean* human judgments (averaged over all participants) for each inference question. For our **extended DKG** domains, we recruit $n = 20$ participants for each variant (totaling 100 participants across 5 variants; mean age = 39.40, 55 female, 45 male). All human data collection took place over a customized web interface (see interface examples in the Appendix), where the participants first completed a tutorial and comprehension check. The experiment is approved by an IRB Board at a US University. Participants were paid $15 USD per hour. We excluded 13 participants who gave high likelihood scores to all options.

3.3 LIRAS Model Configuration

We instantiate the LIRAS model with Gemini 2.0 Flash (Team et al., 2025) as our base VLM for all parsing and code synthesis, and the execution of the inference by SIAM takes place on a MacBook Pro. During code synthesis, we provide the VLM with a generic prompt shared across all domains and variants. This prompt includes a tutorial on SIAM syntax and primitives, and one example toy domain with accompanying parses (see Appendix). We synthesize all code with temp$= 1.0$ to ensure sufficient diversity in initial synthesis, using rejection sampling until we generate a sample for each stimulus in which the full pipeline runs to completion. In our experiments we use $k = 1$ samples per stimulus, as qualitatively semantic variation among models that actually compile is minimal (the fully specified stimuli in the domains we use do not ultimately suggest much uncertainty over environment dynamics – an interesting grounds for future work).

For each stimulus, we provide the model with the full visual input and linguistic experimental setup (including instructions explaining the task, concatenated with the scene scenario and query for each stimulus) shown to human participants. We also augment the instructions to explicitly specify which actions the agent can perform, and the size of the environment grid. These same visual and linguistic specifications are used for all baselines.

3.4 Ablations and Baselines

We compare against the following alternatives:

- **No explicit inference (ablation)**: to probe the role of the explicit Bayesian inference engine (SIAM), we run the LIRAS pipeline to fully synthesize the same symbolic environment and model representations (using the same base Gemini 2.0 Flash model), but then prompt the LLM to directly generate the answers to questions conditioned on the stimulus and generated symbolic model.
- **Chain-of-thought**: we also compare our pipeline to more standard chain-of-thought (Wei et al., 2022) prompting. We evaluate on **Gemini 2.0 Flash** (our base neural model), **GPT-4o** (OpenAI et al., 2024) (an alternate SOTA LM), and **OpenAI o3** (an explicit reasoning model). For all CoT baselines, we use

temp=1 (to match our model) and report average inferences over k=3 samples to better estimate the posterior (our Appendix shows that we find significantly more variability, and worse performance, with any set of just k=1 samples.)

We also experimented with the AutoToM (Zhang et al., 2025) framework, which is explicitly designed for text-based Bayesian Theory of Mind. As AutoToM was only designed for natural language inputs, we experimented with first prompting a VLM (we tried both Gemini 2.0 Flash and GPT-4o) to generate a verbal narrative of the visual stimuli which could be provided to the text-based AutoToM model for end-to-end reasoning. However, we found widespread failure using either base neural model to directly generate an accurate or complete account of the visual inputs in natural language, making downstream text-based reasoning unreliable (see Appendix).

4 Results

LIRAS demonstrates human-like reasoning on social reasoning tasks across domains. We compare LIRAS and baselines against human data. Table 1 and 2 presents the correlations between model predictions and human judgments. We find that LIRAS achieves substantially higher correlation with human responses compared to many alternatives (Table 1). The Gemini 2.0 Flash vanilla model, the foundational visual and parsing component for LIRAS, shows markedly weaker performance (and even several instances of negative correlation with human judgments). One natural question however is whether such models simply cannot reason over visual inputs. We find that the ablated LIRAS model, which synthesizes symbolic world and agent models, but instead uses an LLM to perform probabilistic inference, performs significantly worse than the full model and no better than the Gemini 2.0 Flash base model. This demonstrates the importance of the Bayesian inference engine and highlights that the failure of the smaller VLM models may be beyond visual parsing: even when given a full symbolic representation, they are unable to perform human-like probabilistic social reasoning.

While the state-of-the-art multimodal reasoning model OpenAI o3 exhibits stronger alignment with human judgments than lighter-weight models such as Gemini Flash, its performance still lags significantly behind that of LIRAS and shows a much weaker correlation against human judgments (average $r = 0.63$) on these classical cognitive social reasoning domains. These findings underscore a key limitation of contemporary vision-language models: even when extensively pre-trained and fine-tuned for complex reasoning tasks, they still face challenges in achieving human-like multimodal social reasoning, particularly when prompted to interpret visual scenes conditioned on complex linguistic information about the domain. These results show that grounded Theory-of-Mind reasoning with both language and visual inputs is challenging for most state-of-the-art foundation models – even in classic relatively simple social reasoning domains, echoing recent findings by Buschoff et al. (2025).

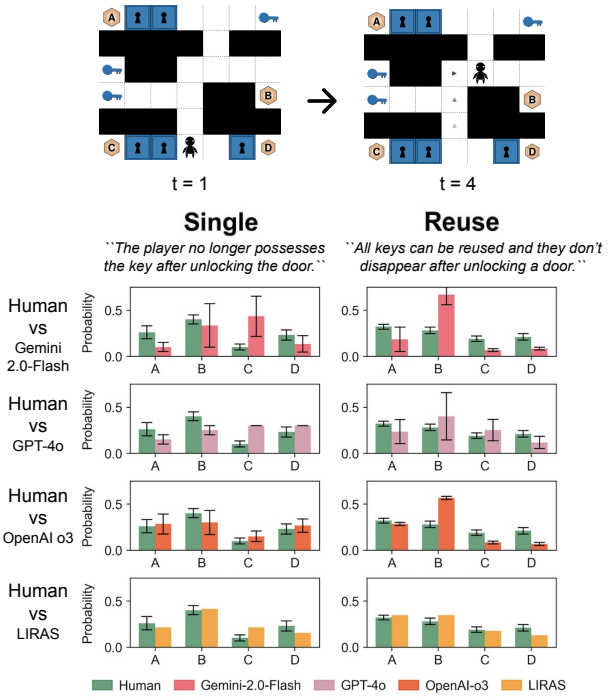

Figure 2: A qualitative example showing how models and human participants adjust their goal inference subject to changes in game dynamics. Top) Two Frames showing the agent walk a few steps up then turn right. Bottom) Model vs Human judgments on the player's goal. In the DKG-Reuse condition where each key can unlock multiple doors, humans and LIRAS find Gem A and B to be almost equally likely, where other baselines generally infer that Gem B is likely the agent's goal.

LIRAS's inference is adaptive and robust to meaningful changes in linguistic inputs. A key motivation for this work is to computationally account for how language shapes inferences about so-

Models	**Foodtruck**		**Astronaut**	
Inference	Belief	Desire	Rewards	Cost
Gemini 2 Flash	0.01 [-0.13, 0.15]	0.23 [0.10, 0.35]	0.11 [-0.15, 0.33]	0.02 [-0.24, 0.23]
GPT 4o	-0.03 [-0.18, 0.12]	0.15 [0.02, 0.26]	-0.20 [-0.42, 0.05]	-0.17 [-0.36, 0.04]
OpenAI o3	0.52 [0.41, 0.62]	0.45 [0.35, 0.54]	0.33 [0.12, 0.49]	0.80 [0.69, 0.88]
LIRAS (Ablated)	0.03 [-0.10, 0.15]	0.19 [0.06, 0.32]	0.20 [-0.04, 0.40]	-0.08 [-0.28, 0.14]
LIRAS (Full)	**0.76 [0.69, 0.82]**	**0.74 [0.68, 0.79]**	**0.87 [0.76, 0.94]**	0.75 [0.63, 0.85]

Table 1: Correlation coefficients and corresponding 95% confidence intervals comparing each model against human judgments on various cognitive domains for social reasoning. DKG results are displayed in Table 2. Scatterplots for each domain are shown in the Appendix.

Models	**DKG-Single**	**DKG-Double**	**DKG-Reuse**	**DKG-Inverse**	**m-DKG**
Gemini 2.0 Flash	0.50 [0.34, 0.64]	0.19 [-0.10, 0.47]	0.21 [-0.01, 0.41]	0.12 [-0.15, 0.38]	0.11 [-0.16, 0.37]
GPT 4o	0.40 [0.24, 0.55]	0.39 [0.07, 0.68]	0.29 [-0.00, 0.58]	0.11 [-0.13, 0.35]	0.36 [0.13, 0.57]
OpenAI o3	0.73 [0.60, 0.83]	0.73 [0.60, 0.83]	0.52 [0.34, 0.69]	0.79 [0.70, 0.87]	0.81 [0.73, 0.88]
LIRAS (Ablated)	0.57 [0.42, 0.69]	0.42 [0.12, 0.66]	0.45 [0.20, 0.66]	-0.11 [-0.31, 0.13]	0.53 [0.33, 0.71]
LIRAS (Full)	0.79 [0.70, 0.84]	0.74 [0.58, 0.83]	0.75 [0.61, 0.83]	0.74 [0.61, 0.86]	0.78 [0.70, 0.85]
Human (Split-half)	0.78 [0.70, 0.84]	0.73 [0.60, 0.84]	0.80 [0.72, 0.88]	0.80 [0.74, 0.86]	0.73 [0.63,0.81]

Table 2: Correlation coefficients for each model on the four variants of the single-agent DKG domain and the multi-agent DKG domain. Overall, LIRAS shows robust correlation against humans across all variants. Other VLM baselines correlate moderately well on the DKG-Single variant, but they are less robust on some other variants with more unusual but interesting dynamics. Model results statistically significant from others are bolded. Scatterplots for each variant are shown in the Appendix.

cial situations. To that end, we next assess LIRAS under variants of the DKG domain that modulate the underlying rule structure, using language. We find in Table 2 that that, similar to the model performances across domains, LIRAS performance is robust across all DKG variants, with correlations roughly at the noise ceiling of the human data (as computed under split-half correlations). We again find that ablations of LIRAS impair performance – as do alternate VLM baselines (Gemini 2.0 Flash and GPT-4o) especially as rules become more unusual. While o3 generally achieves similar performance, we similarly notice a marked drop in performance for the DKG-Reuse condition, where a key can be reused to unlock multiple doors.

To illustrate, we also show a qualitative example in Figure 2. In this example, there are four gems, A, B, C, D. The same visual stimulus were tested under DKG-Single and DKG-Reuse variants. In the Single condition, the OpenAI o3 model finds A and B to be similarly likely, where humans and LIRAS both rate gem B to be more likely. This is because the agent would have gone to get the closer blue key(s) on the left, if they are aiming for gem A, C, or D. Under the DKG-Reuse condition, where the participants are told that each key can be reused after unlocking doors, human judgments change significantly. LIRAS is able to capture this shift in probability distribution, reasoning that now the agent is *equally* likely to go for gem A or gem B, since the top right blue key can now unlock *both* blue doors to get to gem A. On the other hand, all other baseline models judge only gem B to be the most likely option.

4.1 Error Analysis

Overall after inspecting the CoT tokens by the Gemini 2.0 and GPT 4o model, we find widespread hallucination and factual errors in its reasoning. We highlight some in the appendix. The OpenAI o3 model does not output its thinking tokens. We instead ask it to justify its answer with reasoning and we also find factual mistakes and illogical statements even if the final results appear to be similar to human judgments.

In LIRAS, we also noticed some hallucination and syntax errors in world model synthesis and visual parsing. However by giving a structured prompt and parsing the grid cell by cell, LIRAS reduces the inaccuracy in the final synthesized output. Multiple checks and resampling were needed to ensure the model synthesized can be compiled and executed.

5 Discussion

In this paper, we propose LIRAS, a multimodal model synthesis architecture capable of synthesizing agent and world models "on the fly", from

visual and linguistic inputs, to reason about agents' mental states from observation. We test our model on a variety of popular cognitive science domains with different aspects of social reasoning under different settings. Like people, our model flexibly adapts to variations in the underlying rules of the scenario at hand. Our work provides a computational account of how language can help construct an ad-hoc world model for people to contextualize and interpret other agents' behavior.

On the other hand, despite using Gemini 2.0 Flash, a lightweight VLM, in the model architecture, LIRAS is able to match or outperform state-of-the-art multimodal reasoning models on most social reasoning tasks, and significantly exceeds the performance of the same Gemini 2.0 Flash model that LIRAS uses for model synthesis. By constructing these ad-hoc world and agent models on-the-fly, our study is a step towards a generalized cognitive model capable of human-like flexible social reasoning navigating the social world and form more effective human-AI thought partners (Collins et al., 2024).

6 Related Work

6.1 Model-based Theory-of-Mind Reasoning

Our work builds on a long history of research in cognitive science and AI that shows that humans interpret others' behaviors by assuming they are rational agents (Dennett, 1981; Baillargeon et al., 2016). Numerous computational models have been proposed to capture this model-based reasoning process (Baker et al., 2017; Shum et al., 2019; Wu et al., 2021; Alanqary et al., 2021; Ying et al., 2025b) and have been shown to capture graded human judgments in reasoning about agents' mental states from observations.

6.2 Theory-of-Mind in Foundation Models

Theory-of-Mind reasoning in Foundation Models has been subject to great interests and heated debates from the AI and NLP community. Many studies have shown that Foundational Models are capable of human-like Theory-of-Mind reasoning in many real world tasks (Kosinski, 2023), while some have highlighted numerous limitations (Ying et al., 2025a; Ullman, 2023). Recent work has also proposed cognitively inspired approach to teach LLMs to reason about linguistic social scenarios in a Bayesian way through prompting or fine-tuning (Zhang et al., 2025; Kim et al., 2025; Qiu et al., 2025; Zhu and Griffiths, 2024). Recent work (Jin et al., 2024; Shi et al., 2025) has also applied Bayesian Theory of Mind to multimodal settings by having a VLM first converting video to action predicates, although most of such existing work has focused on QA and not capturing graded human uncertainty in Theory-of-Mind reasoning.

6.3 Automated Model Synthesis

The ability for Foundation Models to to synthesize code unlocks new possibilities for automated model synthesis. Automated model synthesis has been applied in different areas, from statistical reasoning (Li et al., 2024; Domke, 2025) and planning (Silver et al., 2023) to cognitive modeling (Wong et al., 2023; Brooke-Wilson, 2023). This preceding work is restricted only to language-based model synthesis, while the current work extends this to the multimodal domain. It is the first work to apply model synthesis to social reasoning, including joint synthesis over world and agent models.

7 Limitations

Our work is not without limitations. First, our current approach is restricted to discrete domains and does not extend to continuous spaces, reflecting a major limitation inherent in the PDDL framework. Additionally, modeling multiagent scenarios remains challenging, particularly in competitive settings; our framework cannot yet adequately capture the complexities of multiagent interactions. While we found that a single set of prompts can handle all four tested domains, the generalizability of this approach to novel domains is not guaranteed, due to possible issues with domain-specific syntactic or semantic mismatches. In addition, the LIRAS model currently parses gridworld domains by enumerating each cell, which does not generalize to more complex visual inputs.

Lastly, our model depends on explicitly provided linguistic information—such as a clearly enumerated action space and well-defined transition dynamics (e.g., specifying that agents cannot walk through buildings). In contrast, humans can often infer such rules implicitly from context, drawing upon commonsense knowledge to build mental models of new environments without explicit instructions.

To address these limitations, future research could focus on scaling up and improving generalization by fine-tuning vision-language mod-

els (VLMs) with more diverse training examples across broader domains. This could enable more robust handling of new task domains on-the-fly. Furthermore, enhancing the model's capacity to infer implicit domain constraints — possibly by incorporating structured priors — could make this approach more generalizable to cases where the linguistic information is ambiguous or incomplete.

References

Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, and 1 others. 1998. PDDL | The Planning Domain Definition language. *Technical Report, Tech. Rep.*

Arwa Alanqary, Gloria Z Lin, Joie Le, Tan Zhi-Xuan, Vikash K Mansinghka, and Joshua B Tenenbaum. 2021. Modeling the mistakes of boundedly rational agents within a bayesian theory of mind. *arXiv preprint arXiv:2106.13249*.

Renée Baillargeon, Rose M Scott, and Lin Bian. 2016. Psychological reasoning in infancy. *Annual review of psychology*, 67:159–186.

Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. 2011. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33.

Chris L Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B Tenenbaum. 2017. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1(4):0064.

Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. 2009. Action understanding as inverse planning. *Cognition*, 113(3):329–349.

Richard Bellman. 1958. Dynamic programming and stochastic control processes. *Information and control*, 1(3):228–239.

Tyler Brooke-Wilson. 2023. *Bounded Rationality as a Strategy for Cognitive Science*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Luca Buschoff, Elif Akata, Matthias Bethge, and Eric Schulz. 2025. Visual cognition in multimodal large language models. *Nature Machine Intelligence*, 7:96–106.

Katherine M Collins, Ilia Sucholutsky, Umang Bhatt, Kartik Chandra, Lionel Wong, Mina Lee, Cedegao E Zhang, Tan Zhi-Xuan, Mark Ho, Vikash Mansinghka, and 1 others. 2024. Building machines that learn and think with people. *Nature Human Behavior*, 8(10):1851–1863.

Logan Cross, Violet Xiang, Agam Bhatia, Daniel LK Yamins, and Nick Haber. 2024. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. *arXiv preprint arXiv:2407.07086*.

Daniel Clement Dennett. 1981. *The Intentional Stance*. MIT Press.

Justin Domke. 2025. Large language bayes. *Preprint*, arXiv:2504.14025.

Carlos Hernández, Xiaoxun Sun, Sven Koenig, and Pedro Meseguer. 2011. Tree Adaptive A*. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 123–130.

Jennifer Hu, Felix Sosa, and Tomer Ullman. 2025. Re-evaluating theory of mind evaluation in large language models. *arXiv preprint arXiv:2502.21098*.

Julian Jara-Ettinger, Hyowon Gweon, Laura E Schulz, and Joshua B Tenenbaum. 2016. The naïve utility calculus: Computational principles underlying commonsense psychology. *Trends in cognitive sciences*, 20(8):589–604.

Julian Jara-Ettinger, Laura E Schulz, and Joshua B Tenenbaum. 2020. The naive utility calculus as a unified, quantitative framework for action understanding. *Cognitive Psychology*, 123:101334.

Chuanyang Jin, Yutong Wu, Jing Cao, Jiannan Xiang, Yen-Ling Kuo, Zhiting Hu, Tomer Ullman, Antonio Torralba, Joshua B Tenenbaum, and Tianmin Shu. 2024. Mmtom-qa: Multimodal theory of mind question answering. *arXiv preprint arXiv:2401.08743*.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Hyunwoo Kim, Melanie Sclar, Tan Zhi-Xuan, Lance Ying, Sydney Levine, Yang Liu, Joshua B Tenenbaum, and Yejin Choi. 2025. Hypothesis-driven theory-of-mind reasoning for large language models. *arXiv preprint arXiv:2502.11881*.

Michal Kosinski. 2023. Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:2302.02083*.

Michal Kosinski. 2024. Evaluating large language models in theory of mind tasks. *Proceedings of the National Academy of Sciences*, 121(45):e2405460121.

Michael Y Li, Emily Fox, and Noah Goodman. 2024. Automated statistical model discovery with language models. In *Forty-first International Conference on Machine Learning*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+p: Empowering large language models with optimal planning proficiency. *Preprint*, arXiv:2304.11477.

Shima Rahimi Moghaddam and Christopher J Honey. 2023. Boosting theory-of-mind performance in large language models via prompting. *arXiv preprint arXiv:2304.11490*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Linlu Qiu, Fei Sha, Kelsey Allen, Yoon Kim, Tal Linzen, and Sjoerd van Steenkiste. 2025. Bayesian teaching enables probabilistic reasoning in large language models. *arXiv preprint arXiv:2503.17523*.

Maarten Sap, Ronan Le Bras, Daniel Fried, and Yejin Choi. 2022. Neural theory-of-mind? on the limits of social intelligence in large lms. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3762–3780.

Luca M Schulze Buschoff, Elif Akata, Matthias Bethge, and Eric Schulz. 2025. Visual cognition in multimodal large language models. *Nature Machine Intelligence*, pages 1–11.

Haojun Shi, Suyu Ye, Xinyu Fang, Chuanyang Jin, Leyla Isik, Yen-Ling Kuo, and Tianmin Shu. 2025. Mumatom: Multi-modal multi-agent theory of mind. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 1510–1519.

Michael Shum, Max Kleiman-Weiner, Michael L Littman, and Joshua B Tenenbaum. 2019. Theory of minds: Understanding behavior in groups through inverse planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6163–6170.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. 2023. Generalized planning in pddl domains with pretrained large language models. *Preprint*, arXiv:2305.11014.

Hao Tang, Darren Key, and Kevin Ellis. 2024. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. *Advances in Neural Information Processing Systems*, 37:70148–70212.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Tomer Ullman. 2023. Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Lionel Wong, Gabriel Grand, Alexander K. Lew, Noah D. Goodman, Vikash K. Mansinghka, Jacob Andreas, and Joshua B. Tenenbaum. 2023. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*, pages arXiv–2306.

Lionel Wong, Jiayuan Mao, Pratyusha Sharma, Zachary S Siegel, Jiahai Feng, Noa Korneev, Joshua B. Tenenbaum, and Jacob Andreas. 2024. Learning grounded action abstractions from language. In *The Twelfth International Conference on Learning Representations*.

Sarah A Wu, Rose E Wang, James A Evans, Joshua B Tenenbaum, and David C et al Parkes. 2021. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432.

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

Lance Ying, Katherine M Collins, Lionel Wong, Ilia Sucholutsky, Ryan Liu, Adrian Weller, Tianmin Shu, Thomas L Griffiths, and Joshua B Tenenbaum. 2025a. On benchmarking human-like intelligence in machines. *arXiv preprint arXiv:2502.20502*.

Lance Ying, Tan Zhi-Xuan, Vikash Mansinghka, and Joshua B Tenenbaum. 2023. Inferring the goals of communicating agents from actions and instructions. *arXiv preprint arXiv:2306.16207*, 2(1):arXiv–2306.

Lance Ying, Tan Zhi-Xuan, Lionel Wong, Vikash Mansinghka, and Joshua B Tenenbaum. 2025b. Understanding epistemic language with a language-augmented bayesian theory of mind. *Transactions of the Association for Computational Linguistics*.

Zhining Zhang, Chuanyang Jin, Mung Yao Jia, and Tianmin Shu. 2025. Autotom: Automated bayesian inverse planning and model discovery for open-ended theory of mind. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Tan Zhi-Xuan. 2022. *PDDL. jl: An Extensible Interpreter and Compiler Interface for Fast and Flexible AI Planning*. Ph.D. thesis, Massachusetts Institute of Technology.

Tan Zhi-Xuan, Nishad Gothoskar, Falk Pollok, Dan Gutfreund, Joshua B Tenenbaum, and Vikash K Mansinghka. 2022. Solving the baby intuitions benchmark with a hierarchically bayesian theory of mind. In *RSS 2022 Workshop on Social Intelligence in Humans and Robots*.

Tan Zhi-Xuan, Gloria Kang, Vikash Mansinghka, and Josh Tenenbaum. 2024a. Infinite ends from finite samples: Open-ended goal inference as top-down bayesian filtering of bottom-up proposals. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 46.

Tan Zhi-Xuan, Jordyn Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. 2020. Online Bayesian goal inference for boundedly rational planning agents. *Advances in neural information processing systems*, 33:19238–19250.

Tan Zhi-Xuan, Lance Ying, Vikash Mansinghka, and Joshua B Tenenbaum. 2024b. Pragmatic instruction following and goal assistance via cooperative language-guided inverse planning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 2094–2103.

Jian-Qiao Zhu and Thomas L Griffiths. 2024. Eliciting the priors of large language models using iterated in-context learning. *arXiv preprint arXiv:2406.01860*.

A Human Data Collection

The interface used for human data collection is shown in Fig. 3. Human participants first complete a consent form and a tutorial for the experiment. They then complete 16 trials of the study in a randomized order. In each trial, they are asked to watch the animation and then rate the likelihood for each goal gem from 0 (Extremely Unlikely) to 100 (Extremely Likely). The results are then normalized as a probability distribution across four ratings such that they sum up to 1. All human subject data collected are anonymized.

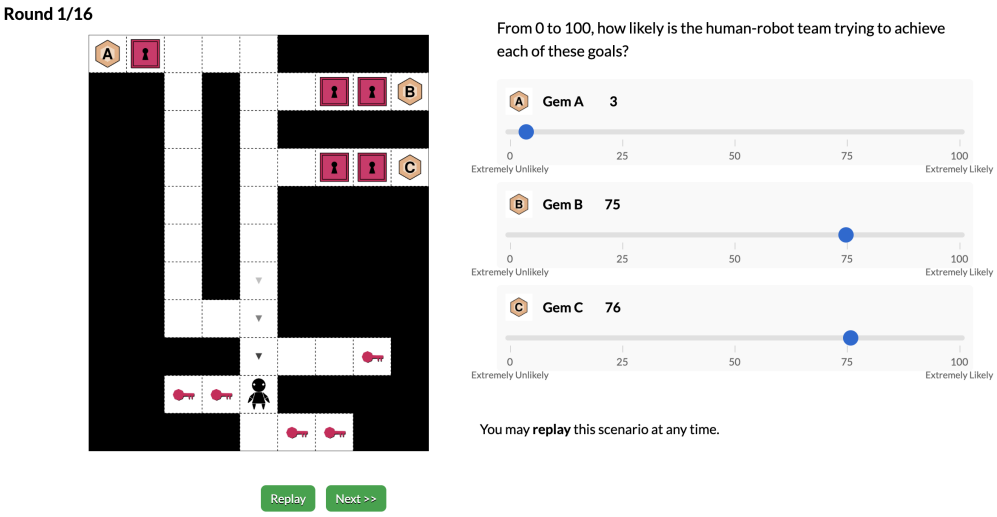

Figure 3: Experiment Interface for the human data collection.

B Translation Accuracy and Runtime Analysis

To assess the deployability of LIRAS, we evaluate both the effectiveness (translation accuracy) and efficiency (runtime) of its neurosymbolic pipeline in the FoodTruck domain.

Accuracy We define translation accuracy as the proportion of synthesis runs that yield a PDDL model passing both syntactic and semantic validation. Under this definition, LIRAS achieves 89.7% accuracy in the FoodTruck domain, indicating that, with Gemini 2.0 Flash as the underlying VLM, a valid PDDL program is produced in 89.7

Runtime We compare end-to-end runtime across models. For baseline VLMs, runtime is measured as the wall-clock time from API request initiation to receipt of the response. For LIRAS, runtime is the total wall-clock time required to execute all stages of the pipeline in Figure 1, including both program synthesis and probabilistic inference.

The results, summarized in Figure **??**, show that non-deliberative models such as Gemini 2.0 Flash (mean = 3.45) and GPT-4o (mean = 20.00) are substantially faster than OpenAI o3 (mean = 39.34). LIRAS

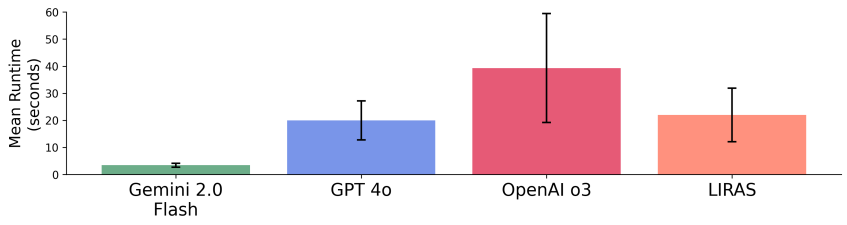

Figure 4: Comparison of runtime across models. LIRAS takes longer to run than Gemini 2.0 Flash but significantly faster than OpenAI o3.

(mean = 22.03) incurs additional overhead relative to Gemini 2.0 Flash but remains markedly faster than o3. Because LIRAS inference was executed on a personal MacBook, further speedups are likely with optimized hardware and parallelization.

C Analysis of Qualitative Examples

Snippets of Gemini 2.0 Flash reasoning on the DKG-Reuse condition for the example highlighted in Fig. 2.

> **Gem-A**: Far from the starting position, requiring navigation across the locked doors. No indication of movement in that direction so far. **Gem-B**: Accessible directly without requiring a key. The player didn't move toward it, even though it's relatively close. **Gem-C**: Behind a door that requires a blue key. The player hasn't moved toward a key to unlock it. **Gem-D**: Also behind a door requiring a blue key. Similar to Gem-C, no movements have been made toward unlocking it.

Snippets of GPT 4o 2.0 reasoning on the DKG-Reuse condition for the example highlighted in Fig. 2.

> The player is likely trying to collect gem B as there is no way for the player to reach gem A or D without going to the right, and the player is heading towards that side. Alternatively, the player could be heading to get the key to unlock gem D or C.But, the key on the right is more accessible. So, it is more likely that the player is going to collect gem B. [5, 80, 5, 10]

Snippets of OpenAI o3's reasoning on the DKG-Reuse condition for the example highlighted in Fig. 2.

> • Gem B (right-centre): From the current square the quickest, completely key-free way to a gem is to walk right through the top of the vertical gap (col 6) and then drop down to gem B. This route is open and costs only a few steps, so B is the most natural target.
>
> • Gem A (top-left): To obtain A the player must – keep moving right to the column-6 opening, – climb to the top row, – walk all the way to the blue key at the far right, – return left and unlock two blue doors, then collect A. The present position is consistent with the first of those steps, but the plan is quite a bit longer and key-dependent.
>
> <answer>[10,60,5,25]</answer>

D Representing Spatial Input for Text-Based Reasoning Models

We also attempted to compare LIRAS against AutoToM (Zhang et al., 2025) a recent work which similarly combines LLM-guided model synthesis with Bayesian inference for social reasoning. However, AutoToM is restricted to textual input. Our stimuli here critically revolve around rich visual input extended over multiple time steps. While we attempted to automatically convert the GIFs to text-only narratives describing the scene, we found that the descriptions were often impoverished and not giving AutoToM a fair chance. We show a few example narratives to demonstrate the challenges. We do so not to illustrate that current models are incapable of describing such scenarios but to highlight that baseline use for such visual description is nontrivial.

We tried both Gemini 2.0 Flash and GPT-4o. We prompted Gemini through the API, concatenating png slices of the GIF; we used the default temperature 1.0. We separately prompted GPT-4o through the chat interface (uploading the GIF directly). In both cases, we described the Doors, Keys, Gems game environment and task the models with describing the scene in detail (where objects were and what the agent did). We depict example representative narrative descriptions for the figure we show in Figure 2.

Gemini:

The player begins in the middle of the screen. The player moves down one square. Then the player moves up one square.

GPT-4o:

The player starts near the center of the grid. They move up one space to get a better view. Spotting a blue key to the left, they move left and pick it up. With the first key in hand, the player returns to the center and heads right. They acquire a second blue key on the right side.

E Sequential Inverse Agent Modeling

In Algorithm 1, we provide pseudo-code for the Sequential Inverse Agent Modeling (SIAM) algorithm introduced in Section 2.4. We show the case where all possible mental states we consider (goals, rewards, costs, and beliefs) are jointly inferred, since dropping any of the mental states corresponds to a special case.[2] We note that since we consider deterministic PDDL environments, the expected utility of reaching a goal g is equivalent to the goal reward r_g plus the cost of the shortest path of reaching that goal. The cost of the shortest path PATH-COST can be efficiently computed via A* search, with the results memoized. Shortest path computations can even be incrementally computed using approaches like tree-adaptive A* (Hernández et al., 2011), such that once a shortest path is found from state s, finding a shortest path from a nearby state s' is often much cheaper. We refer the reader to the appendices of Ying et al. (2025b) and Zhi-Xuan et al. (2024b) for more explanation of how memoization and incremental planning can be exploited by SIPS-derived algorithms like SIAM.

Algorithm 1 Sequential Inverse Agent Modeling (SIAM) for mental state inference

procedure SIAM($\mathsf{G}, \mathsf{R}, \mathsf{C}, \mathsf{B}_0, a_{1:T}, s_{0:T}$)
  $\mathcal{H} \leftarrow \mathsf{G} \times \mathsf{R} \times \mathsf{C} \times \mathsf{B}_0$ ▷ Enumerate all hypotheses (goal, reward, cost, & belief combinations).
  $\mathcal{W} \leftarrow \{w^i := P(g^i|r^i, c^i, s_0)\}_{i=1}^{|\mathcal{H}|}$ ▷ Initialize (unnormalized) weights for all hypotheses.
  for $t \in [1, T]$ **do**
    for $h^i := (g^i, r^i, c^i, b^i_{0:t-1}) \in \mathcal{H}$ **do**
      $b^i_t \leftarrow$ BELIEF-UPDATE$(b^i_{t-1}, s^i_t, a_{t-1})$ ▷ Simulate agent's belief update.
      $h^i \leftarrow (g^i, r^i, c^i, b^i_{0:t})$
      $Q_{\text{Bel}}(g^i, r^i, c^i, b^i_t, \tilde{a}) \leftarrow 0$ **for** $\tilde{a} \in$ VALID-ACTIONS(b^i_t) ▷ Initialize belief-space Q-values.
      for $(\tilde{s}, \tilde{w}) \in b^i_t$ **and** $\tilde{a} \in$ VALID-ACTIONS$(\tilde{s})$ **do** ▷ Iterate over environment states in agent's belief.
        $Q^*(g^i, r^i, c^i, \tilde{s}, \tilde{a}) \leftarrow$ MEMOIZED(PATH-COST$(\tilde{s}, \tilde{a}, g^i, r^i, c^i) + r_{g^i}$) ▷ Compute shortest path cost to g.
        $Q_{\text{Bel}}(g^i, r^i, c^i, b^i_t, \tilde{a}) \leftarrow Q^*(g^i, r^i, c^i, b^i_t, \tilde{a}) + \tilde{w} \cdot Q(g^i, r^i, c^i, \tilde{s}, \tilde{a})$ ▷ Update belief-space Q-values.
      end for
      $P(a_t|b^i_t, g^i, r^i, c^i) \leftarrow \frac{\exp(\beta Q_{\text{Bel}}(g^i, r^i, c^i, b^i_t, a_t))}{\sum_a \exp(\beta Q_{\text{Bel}}(g^i, r^i, c^i, b^i_t, a))}$ ▷ Compute likelihood of action a_t.
      $w^i \leftarrow w^i \cdot P(a_t|b^i_t, g^i, r^i, c^i)$ ▷ Update weight with action likelihood.
    end for
  end for
  return $(\mathcal{H}, \mathcal{W})$ ▷ Return all hypotheses and their (unnormalized) weights.
end procedure

F LIRAS Model Synthesis Prompts

F.1 Prompt for synthesizing the PDDL domain model

```
"""
You will read instructions about a game setup and a dictionary of objects presented
in the problem. You will then synthesize a PDDL domain for the text description you are
given.

Note that in our PDDL definition, we use a bit-matrix and array to represent different types
of cells. These cell types generally refer to generic barriers or kinds of terrains / spaces.

If the same actions have different costs depending on the cells it is located, then each type
of cell should have a separate action definition. Note that the costs of actions on each
terrain will be represented in a separate file and you do not need to encode that in the PDDL
domain file. If certain cells represent barriers, then make sure you cannot move onto those
cells.
```

[2] Modulo minor differences — when goal rewards R are not specified, goals are assumed to be distributed from a uniform prior. When rewards are specified, the agent is modeled as initially selecting a goal from a Boltzmann distribution over the net utility of each goal (i.e. goal reward minus shortest path cost to goal), as in Jara-Ettinger et al. (2020).

```
You must include all generic_objects from the object dictionary in the PDDL types.
The predicates should be about the states, relations or attribute of objects
(e.g. you can define isshape, iscolor, isempty, etc.). Please do not include predicates
that are not relevant for the agents' goals (i.e. don't represent every possible
object attribute). The types in the PDDL domain definition should refer to broad
category of objects (e.g. fruit) and attributes (e.g. shape, color) and not specific instances.

The task instructions will provide you with a list of actions that can be taken by the agent(s).
Please do not invent any new actions in the PDDL domain file.

Here is an example:

Input: In this domain, you are observing a boy trying to reach some balls and plates.
There are three unique balls: a tennis ball, a basketball and a baseball.
The plates can have shapes of circle or square. The plates are placed inside cabinets.
The agent can move up, down, left or right. There are whitespaces and blackspace in the map.
The agents and objects can only exist in whitespaces.

objects = {'generic_objects': ['ball', 'plate', 'cabinet'],
'unique_objects': ['tennisball', 'basketball', 'baseball'],
'background_cells': ['whitespace', 'blackspace'],
'agent': ['boy']}

Output:

(define (domain example)
    (:requirements :fluents :adl :typing)
    (:types
        ball plate  - item ; you may include small generic objects
        item cabinet agent - object ; include 'item', 'agent' and any other objects
        shape ;this can be shape, color or other attributes
    )
    (:predicates
        (has ?a - agent ?i - item)
        (at ?a - agent ?o - object) ; do not change
        (adjacent ?a - agent ?o - object) ; do not change
        (isplateshape ?p - plate ?s - shape)
        (isballshape ?b - ball ?s - shape)
    )

    (:constants
        boy - agent ; name(s) of the agent(s) should be listed here,
        circle square - shape ; list kinds of attributes mentioned
        tennisball basketball baseball - ball ; list all unique objects
    )

    (:functions
        (gridheight) - integer
        (gridwidth) - integer
        (xloc ?o - object) (yloc ?o - object) - integer
        (whitespace) (blackspace) - bit-matrix ;
        this should be an exact list as in physical_generic_objects["background_cells"]
    )

    (:derived (at ?a ?o) (and (= (xloc ?a) (xloc ?o)) (= (yloc ?a) (yloc ?o))))

    (:action pickup
     :parameters (?a - agent ?i - item)
     :precondition
        (and (not (has ?a ?i))
            (adjacent ?a ?i)
     :effect
        (and (has ?a ?i)
        (assign (xloc ?i) -1) (assign (yloc ?i) -1)
        )
    )
    )

    (:action up-white
     :parameters (?a - agent)
     :precondition
        (and (> (yloc ?a) 1)
            (= (get-index whitespace (yloc ?a) (xloc ?a)) true)
            (= (get-index blackspace (- (yloc ?a) 1) (xloc ?a)) false)
        )
     :effect
        (and (decrease (yloc ?a) 1))
    )

    [omitting other actions]

)

In cases where you can move on the black space but at a different cost:

    (:action right-white
```

```
     :parameters (?a - agent)
     :precondition
        (and (< (xloc ?a) (gridwidth))
            (= (get-index whitespace (yloc ?a) (xloc ?a)) true)
        )
     :effect
        (and (increase (xloc ?a) 1))
    )

    (:action right-black
     :parameters (?a - agent)
     :precondition
        (and (< (xloc ?a) (gridwidth))
            (= (get-index blackspace (yloc ?a) (xloc ?a)) true)
        )
     :effect
        (and (increase (xloc ?a) 1))
    )

Multiagent cases:

    If multiple agents are present, we use an agent-code to number the agents
    and a turn variable to indicate which agent is in turn to act.

    Include these in the functions only in cases with more than 1 agent:

            (agentcode ?a - agent) - integer
            (turn)- integer

    then you should check (= turn (agentcode ?a)) as a precondition for each agent's action.
    Then after completing the action, we would move on to the next agent:  (assign turn (- 1 turn))


Now please generate a PDDL domain given the input below:

"""
```

F.2 Prompt for Parsing Image Cells

```
Your task is to take an image of a cell in a gridworld, then output a json file describing the object
in the cell, using pddl, based on the pddl domain definition.

There can be multiple objects in the cell, please make sure you represent them all.

The location should be set with xloc and yloc, using placeholder $i and $j for the values.
Please also remember to add relevant attributes such as color and shapes if applicable.

Example:

Input:

Instruction: There are pins and balls on the table. The items can be used or new.
The pins can have a circle or square shape.


object_types:
{
    "generic_objects":  ["pin"],
    "unique_objects": ["tennisball", "basketball", "baseball"],
    "agent": ["human"]
}

pddl_predicates: (isShape ?p - pin ?s - shape) (isNew ?i - item)


Image: [insert image of the cell showing a new circle pin and a baseball on a white square]

Output:

{
"object_name": ["pin", "baseball"]
"object_pddl_str": "(= (yloc pin) $i) \n(= (xloc pin) $j) \n(isShape pin circle) \n(isNew pin) \n(=
(yloc baseball) $i) \n(= (xloc baseball) $j)\n "
}

if you are classifying objects that have attributes such as shapes and colors, you must include those attributes.
```

F.3 Prompt for Synthesizing the Agent Configuration

```
Your task is to synthesize a json configuration file given the problem description.

Grid_size: size of the grid, row by column.

Observability: should either "full" or "partial", indicating whether the agent can see the full map.

Belief config: If observability is full, return an empty dictionary, otherwise,
```

include information below. Note that you should read these from

1. Belief_object: In case of partial observability, you should indicate what is the item being hidden.

2. Belief_container: In case of partial observability, you should indicate what are the containers
   for the hidden objects.

3. Barrier: Name any physical barrier that obstructs the view of the agent.

4. Agent: Name of the agent.

Goals: should be a list of predicate strings. If the problem doesn't call for goal inference
(i.e. goal is given), then this should be a list with 1 predicate. Please note that sometimes the goal
can be a composite if the agent can have multiple objectives. For example, if an agent's goal is to get to get
home, but has the option to pick up a flower or pizza on the way home, the goal space would be [["(at agent home)"],
["(at agent home)", "(has agent flower)"], ["(at agent home)", "(has agent pizza)"], ["(at agent home)", "(has agent flower)",
"(has agent pizza)"]]. Make sure the predicates are allowed based on the PDDL domain definition. Make sure all goal object
names match with the object names provided to you.

Costs: should be a list of possible different action cost profiles (dictionary). The action names should match exactly
with the actions from the PDDL description file. If the task doesn't call for different action costs (i.e. action costs vary
across different cells), then this should have only 1 action cost profile. Action costs should be a real number greater than 0.
In general, specific actions such as pickup should have higher costs than movements.

Query: should be one or more of the following: "belief", "goal", "reward", "cost". Note that rewards usually asks how much does agent
like X, whereas goals questions ask which item is the agent's goal.

Temperature: How rational is the agent? The value should be a real number greater than 0. A lower temperature indicates more rational
actions. By default this should be 1.

==========

Here is an example:

Suppose the task is to infer a human's beliefs, goals, as well as the costs of movement in two kinds of terrains, black and
white, at a 3 by 4 grid. The human cannot see through the black terrain.

The human's goal is to get one of the three balls: baseball, basketball, and tennisball. The balls are hidden in boxes and the agent
cannot see which ball is in which box.

actions from PDDL domain: up-white, down-white, left-white, right-white, up-black, down-black, left-black, right-black, pickup

object types:

```
{
    "generic_objects": ["ball", "box"],
    "unique_objects": [
        "baseball", "basketball", "tennisball"
    ],
    "agent": [
        "human"
    ]
}
```

Example output:

```
{
    "grid_size": [3,4],
    "observability" : "partial",
    "belief_config" : {
        "belief_object": "ball",
        "belief_container": "box",
        "barrier": "blackterrain",
        "agent" : "human"
    },
    "goals": [["(has human baseball)"], ["(has human basketball)"], ["(has human tennisball)"]],
    "costs": [
        {
            "up-white": 1, "down-white": 1, "left-white": 1,  "right-white": 4, "up-black": 4, "down-black": 4,
            "left-black": 4,  "right-black": 4, "pickup": 5,
    }
        {
            "up-white": 2, "down-white": 2, "left-white": 2,  "right-white": 2, "up-black": 2, "down-black": 2, "left-black":2,
            "right-black": 2, "pickup": 5,
    }
        {
            "up-white": 4, "down-white": 4, "left-white": 4,  "right-white": 4, "up-black": 1, "down-black": 1, "left-black": 1,
            "right-black": 1, "pickup": 5,
    }

    ],
    "query":["belief", "goal", "costs"],
}
```

Now please generate a configuration file for the following scenario:

G Scatterplots for Model and Human Judgments

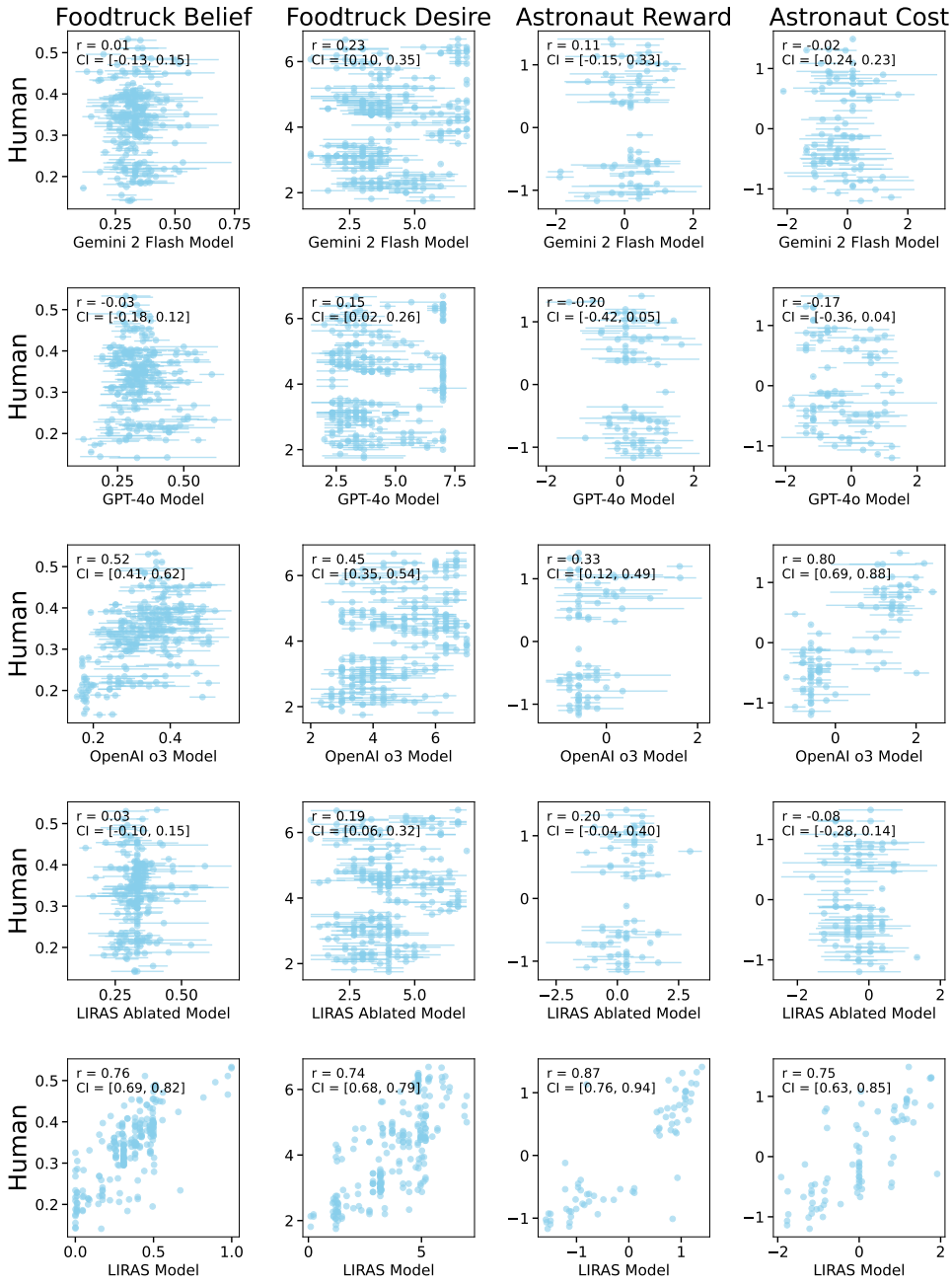

Figure 5: Scatterplots of model vs human judgments on Foodtruck and Astronaut domains. Error bar indicates standard deviation.

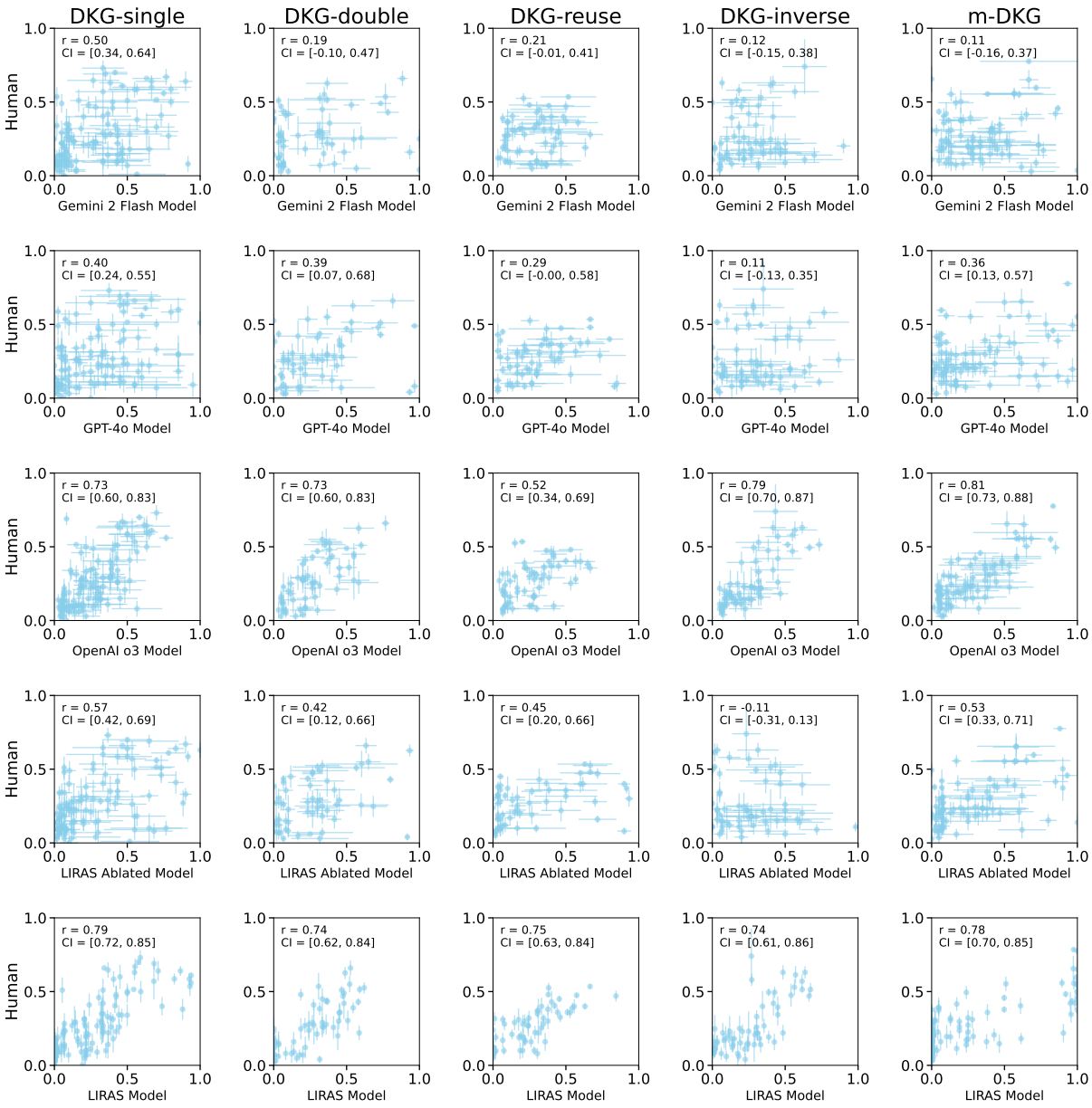

Figure 6: Scatterplots of model vs human judgments on DKG domains and variants. Error bars indicate standard deviation.