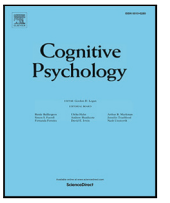
Cognitive Psychology 138 (2022) 101509

Contents lists available at ScienceDirect

Cognitive Psychology

journal homepage: www.elsevier.com/locate/cogpsych



Inductive biases in theory-based reinforcement learning

Thomas Pouncy [a,*], Samuel J. Gershman [a,b]

[a] Department of Psychology and Center for Brain Science, Harvard University, United States of America
[b] Center for Brains, Minds and Machines, MIT, United States of America

ARTICLE INFO

Keywords:
Cognitive science
Psychology
AI
Learning
Inductive bias

ABSTRACT

Understanding the inductive biases that allow humans to learn in complex environments has been an important goal of cognitive science. Yet, while we have discovered much about human biases in specific learning domains, much of this research has focused on simple tasks that lack the complexity of the real world. In contrast, video games involving agents and objects embedded in richly structured systems provide an experimentally tractable proxy for real-world complexity. Recent work has suggested that key aspects of human learning in domains like video games can be captured by model-based reinforcement learning (RL) with object-oriented relational models—what we term *theory-based RL*. Restricting the model class in this way provides an inductive bias that dramatically increases learning efficiency, but in this paper we show that humans employ a stronger set of biases in addition to syntactic constraints on the structure of theories. In particular, we catalog a set of semantic biases that constrain the content of theories. Building these semantic biases into a theory-based RL system produces more human-like learning in video game environments.

1. Introduction

In realistically complex environments, learning is impossible without inductive bias: the set of plausible hypotheses needs to be restricted in some way to ensure that the correct hypothesis can be identified from finite data. This point, which has been formalized in several ways (Gold, 1967; Valiant, 1984; Vapnik & Chervonenkis, 1971; Wolpert, 1996), imposes fundamental design principles on intelligent systems, both artificial and natural. Loosely speaking, such systems must be able to make an educated guess about the correct hypothesis before seeing data. As the philosopher Bertrand Russell put it, "A mind perpetually open will be a mind perpetually vacant."

Since humans are capable of learning in many complex environments, one of the central questions of cognitive science is what kind of inductive biases empower us to do this (Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010). Over the years, researchers have provided myriad answers to this question. For example, human category learning is consistent with a bias for categories defined over single (Medin, Wattenmaker, & Hampson, 1987), separable feature dimensions (Nosofsky, 1987). Research on function learning has shown that humans are biased towards learning linear (Kalish, Lewandowsky, & Kruschke, 2004), simple (Little & Shiffrin, 2009), and compositional functions (Schulz, Tenenbaum, Duvenaud, Speekenbrink, & Gershman, 2017), and in decision making tasks, humans assume that future reward is related to recent context (Gershman & Niv, 2015), and that strategies from previous tasks often generalize to similar tasks (Tomov, Schulz, & Gershman, 2020).

Much of what we know about human inductive biases comes from studying relatively simple task domains. While these simple domains have been useful in identifying many elements of human cognition, they fail to capture some key components of real-world learning environments. Everyday decision making tasks often involve many hidden, intertwining systems: Physical, biological,

* Corresponding author.
E-mail address: pouncy@g.harvard.edu (T. Pouncy).

https://doi.org/10.1016/j.cogpsych.2022.101509
Received 4 January 2022; Received in revised form 16 July 2022; Accepted 23 August 2022

psychological, and social systems link together to produce complex sequential decision problems, and it has been argued that biases relating to these systems are integral to human learning (Lake, Ullman, Tenenbaum, & Gershman, 2017). The current work seeks to better understand human learning in these complex domains.

Video games introduce some key elements of real-world complexity while remaining tractable from a modeling standpoint. Much like the real world, video games can involve physical systems of varying complexity, rich taxonomic relations between objects, agents with various degrees of psychological nuance, and implicit social norms. Yet unlike the real world, video games allow us to both control the complexity of these systems and know the true value of many of their hidden states. This ground truth knowledge of the environment can be fed directly into models in order to avoid having to provide an end-to-end account of cognition for every study. By finding the right combination of task complexity and knowledge representations, we can gradually scale up existing models of human decision making to realistic domains.

Prior work has found that human decision making in simple domains is well described with a mix of model-free and model-based reinforcement learning (RL; Daw, Gershman, Seymour, Dayan, & Dolan, 2011; Daw, Niv, & Dayan, 2005; Kool, Cushman, & Gershman, 2018). Efforts to scale model-free and model-based RL systems up to more complex domains have primarily focused on embedding biases for particular kinds of representational structures. For example, RL systems that learn representations involving structural elements like objects, relations, and rules learn more quickly and robustly in complex domains (Higgins et al., 2018; Pasula, Zettlemoyer, & Kaelbling, 2007; Zambaldi et al., 2018). While adding these structural biases to model-free RL has so far failed to produce human-like learning efficiency (Tsividis, Pouncy, Xu, Tenenbaum, & Gershman, 2017), integrating them into model-based approaches has been more successful. In recent work we showed that combining theories containing object-oriented, relational rules with model-based RL produces human-level learning efficiency and generalization behavior in video game domains (Pouncy, Tsividis, & Gershman, 2021; Tsividis et al., 2021). We refer to this combination of theory-based representations and model-based RL as *theory-based RL*.

Restricting the model class to object-oriented relational theories can be understood as a kind of syntactic inductive bias: only models that have a specific syntactic structure are considered by the learning system. However, prior work suggests that humans also exhibit a number of biases about the semantic content of learned representations. For example, humans make use of notions like design, purpose, and intention to learn categories and infer object usage (Bloom, 1996; Dennett, 1987; Duncker, 1945; Kelemen & Carey, 2006). Humans also learn preferences for certain kinds of elegant solution structures in domains like mathematics (Schoenfeld & Herman, 1982), programming (Weiser & Shertz, 1983), and physics (Chi, Feltovich, & Glaser, 1981). Though it is likely that these kinds of biases also play a role in sequential decision problems, this has not yet been formalized or systematically studied.

In summary, there is good reason to think that theory-based RL provides a plausible account of the representational structure that humans use to learn in complex sequential decision making domains like video games. However, we know relatively little about the set of inductive biases that humans employ to make learning these representations tractable. The first goal of the current study is to address this gap by collecting systematic data about human biases in video game learning. The second goal of the study is to show how these biases can be formalized within a theory-based RL system that learns and acts in human-like ways.

2. Theoretical framework

2.1. An overview of theory-based RL

The general challenge of sequential decision making lies in identifying which sequences of actions are most likely to lead to a desired set of goals. RL approaches solve this problem by translating information about goals into a reward function and learning a value function that tracks the expected future reward associated with the actions available in each task state. As these value functions implicitly encode information about both the goals and dynamics of the environment, they may need to be updated as new information becomes available. Model-based RL addresses this issue by using an internal model of the environment to re-estimate the value function when needed.

While internal models can take many forms, they are often represented as look-up tables that map task states and actions to probability distributions over subsequent states. Tabular models are easy to describe and learn, but they do not scale well to video game environments. The combinatorial structure of objects and interactions produces a large state space that would be intractable to store in a single table. This limitation has stimulated the development of object-oriented and relational representations for model-based RL (Diuk, Cohen, & Littman, 2008; Guestrin, Koller, Gearhart, & Kanodia, 2003; Kansky et al., 2017; Lang & Toussaint, 2010; Pasula et al., 2007; Scholz, Levihn, Isbell, & Wingate, 2014). A key advantage of such approaches is that they express task knowledge compactly. For example, a game like chess has a vast number of distinct states (board configurations), making a tabular approach impractical. States can be represented much more compactly by expressing them in terms of each piece's position on the board. The dynamics of the game can then be expressed in terms of changes in piece position. Theory-based RL takes this a step further, extracting core knowledge about objects, events, and goals into a separate reasoning engine. This engine then turns general knowledge into specific predictions much in the same way that a physics student can apply a general understanding of gravity to predict the path of a specific falling apple depending on its starting height and mass. Thus, fairly complex tasks can be represented with compact theories consisting solely of object properties, interaction rules, and goal information. The compactness of theory-based representations is one ingredient in the recipe for human learning efficiency. Another ingredient – the main focus of our study – is the set of structured inductive biases that restrict the space of plausible theories. Below we will present a general and powerful way to express inductive biases over object-oriented relational theories.

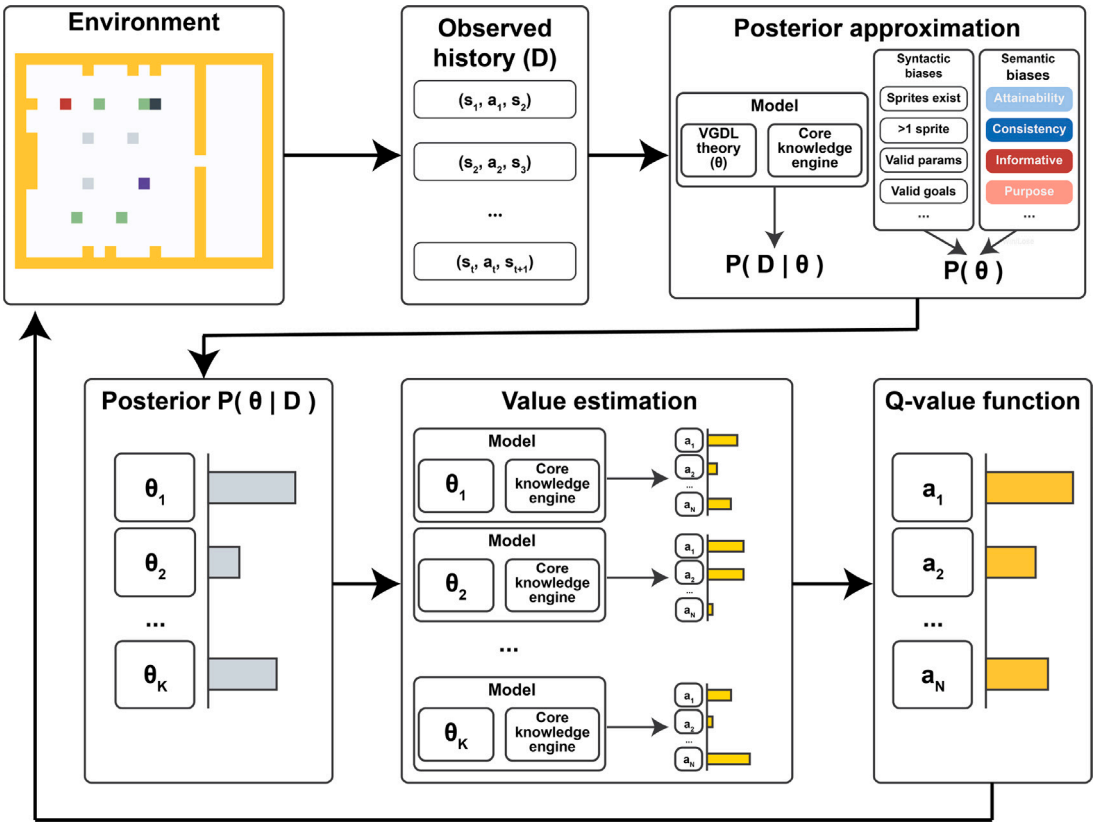


Fig. 1. Schematic of the theory-based reinforcement learning architecture. Agents observe state transitions from the environment then infer a posterior over video game description language (VGDL) theories by combining likelihood estimates from a core knowledge engine and prior probabilities from a series of bias functions. One set of bias functions captures the syntax of the VGDL theory language, while the other set of biases represent biases about the semantic content of theories. Agents use the approximate posterior to estimate a value function and select optimal actions. Gray bars represent probabilities; yellow bars represent expected future reward.

The major computational steps in theory-based RL are summarized in Fig. 1. Game play data is used to compute an approximate posterior distribution over theories, which incorporates knowledge about the core knowledge engine (i.e., the mapping from theory to game play data) and inductive biases about theories. Conditional on a hypothetical theory, a planner uses the core knowledge engine to estimate action values and then select an action. We elaborate each of these steps in the following sections. Further computational details can be found in Appendix A.

2.2. Inferring theories from observation

The central question of theory inference is how people use observations of their environment to infer complete theories about how that environment works. Intuitively, the posterior probability of a complete theory θ given a set of observations $\mathbf{D}$, $P(\theta|\mathbf{D})$, should be related to how well the theory matches the data, $P(\mathbf{D}|\theta)$, and the prior probability of the theory, $P(\theta)$. Bayes' rule formalizes this relationship:

$$P(\theta|\mathbf{D}) \propto P(\mathbf{D}|\theta)P(\theta), \tag{1}$$

The probability of a set of observations $\mathbf{D}$ given a particular theory θ should be higher when a theory can explain all the observations than when it can only explain some. Here we will rely on a common sequential decision making formalization and represent $\mathbf{D}$ as a set of observed state transitions. In other words, each element in $\mathbf{D}$ is a tuple (s, a, s') representing an observation of a player taking an action a from state s and ending up in state s'. We can then formally represent the likelihood of a set of observations given a particular theory $P(\mathbf{D}|\theta)$ as follows:

$$P(\mathbf{D}|\theta) = \prod_{s,a,s' \in \mathbf{D}} P(s'|a, s, \theta), \tag{2}$$

We can calculate the transition probability $P(s'|a, s, \theta)$ by plugging θ into the core knowledge engine to query a model of the environment. In general, theory-based RL is agnostic to the exact language used to encode theories and the core knowledge engine.

For practical purposes, we used a subset of the Video Game Description Language (VGDL) as our theory language (Schaul, 2013). VGDL is a rule-based task representation designed to describe a wide variety of 2D video game tasks using a small fixed ontology. Each theory θ consists of a VGDL description containing three components: a sprite set, an interaction set, and a termination set. The sprite set describes the kinds of objects present in a game and the properties of those objects. The interaction set describes the outcome of an interaction between any two kinds of objects. The termination set describes the goals of the game (i.e., how the player wins or loses). Fig. 2b shows an example of a VGDL description with all three components. We model the core knowledge engine with a VGDL parser that takes as input a VGDL theory, a game state, and a proposed action, and outputs a probability distribution over subsequent game states $P(s'|s,a,\theta)$.

In the interest of tractability, we focused on a subset of deterministic, grid-based VGDL games involving a single moving avatar.[1] Focusing on deterministic VGDL theories means that the transition probability is zero for any transitions other than the one predicted by a given theory. We can then represent the game dynamics in terms of a deterministic transition function $s' = T(s,a)$. For inference purposes this would mean that theories that predict none of the observations would be just as likely as theories that predict every state transition except one. Thus, during theory inference we instead use a "relaxed" formulation of the deterministic transition function that allows for some gradations when theories do not fully explain the observations:

$$P^*(s'|a,s,\theta) \propto \exp(\mathbb{1}[s' = T(s,a)]), \tag{3}$$

where the indicator function takes the value $\mathbb{1}[\cdot] = 1$ if its argument is true, otherwise 0. Taking the exponential of this Boolean indicator function ensures that the transition probability is always non-negative. See Appendix A for additional details about the likelihood function implementation.

2.3. A structured prior over theories

Our primary focus in this work is on the prior probability distribution $P(\theta)$. This prior encodes an agent's inductive biases about what makes one theory more plausible than another in the absence of any observed data. One way to think of these inductive biases is as statements about the structure or content of plausible theories. For example, an agent that holds the bias "a game theory should contain at least one way to win" should find a theory that has at least one win condition more *a priori* plausible than a theory with no win conditions. Formally, if each bias B_i is a Boolean function that operates over theories, $B_i(\theta) \in \{0,1\}$, then theories where $B_i(\theta)$ is true should be more likely *a priori* than theories where $B_i(\theta)$ is false. This intuition suggests that if we had access to an agent's complete set of biases about theories $\mathbf{B}$, the prior probability $P(\theta)$ should be greater for theories where more biases are true than theories where fewer biases are true. For practical purposes, just as with the transition probability used in the likelihood function, our prior over theories should not go to zero if a theory fails to meet any individual bias. We can formalize a prior that satisfies these properties as follows:

$$P(\theta) \propto \exp\left[\sum_i \omega_i B_i(\theta)\right], \tag{4}$$

Here ω_i represents a bias weight for the ith bias function that allows some biases to affect a theory's plausibility more strongly than others. This prior has the important property that the probability of a theory declines incrementally when a bias is violated; thus, the biases act as "soft constraints" on the theory.

This theory prior is a form of *Markov logic network*, which has been widely used in probabilistic logical modeling (Richardson & Domingos, 2006). One way to think about this model is that the bias weights represent the log-odds of each bias being true, holding the truth value of all other biases fixed.[2] The probability distribution over theories can then be understood as the distribution with maximum entropy subject to the constraint that the log-odds of each bias is equal to its bias weight. Intuitively, this means that the theory prior is as weak as possible, while still obeying a specific set of structured inductive biases.

2.4. Specifying the theory space

To produce a precisely defined theory space, we developed a method of converting VGDL theories into a numeric representation. We do this by defining a set of theory predicates that each relate to a specific part of a theory. We can then enumerate the possible values for each theory part and produce a predicate/value representation for the theory as a whole. Fig. 2b shows an example VGDL theory, while 2c shows the set of predicate/value pairs that would describe this theory. The first line of the termination set for this theory indicates that destroying all (i.e., "limit=0") of the green blocks (i.e., "s(prite)type=green") will win (i.e., "win=T") the game. This first termination condition corresponds to the "HasCondition(1)", "Stype(1)", "Limit(1)", and "Win(1)" predicates in Fig. 2c, where the "1" indicates that we are referencing the presence, sprite type, limit, and win value of the first line of the termination set. The specific values of these predicates indicate that the first goal is present, involves the fourth sprite type in the sprite set, is triggered when there are zero of this sprite type left, and results in winning the game, respectively. We can now formalize the notion of "the space of possible theories" more precisely. Since each set of predicate/value pairs maps to a single theory, the space of all possible predicate/value pairs implicitly defines a space of possible theories.

[1] It is worth noting that given the compositional nature of VGDL, even this restricted theory space is quite large.

[2] In general, it may not be possible to hold the truth values of other biases fixed, but this assumption only serves an explanatory purpose; it is not invoked in our modeling.

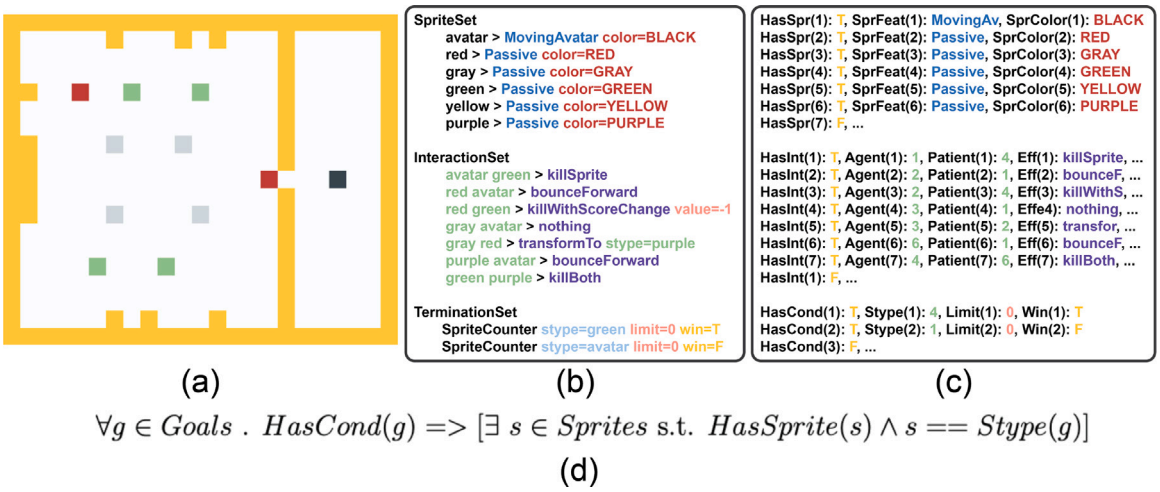


Fig. 2. A visualization of converting VGDL theories to a predicate/value representation to produce a space of possible theories. (a) A screenshot of an example game (referred to as the "Fill the gaps" game in the text). (b) The VGDL description for the game in (a). Different colors represent types of theory elements (e.g., pink represents integers, yellow represents Booleans, dark red represents color values, green represents sprite names). (c) The same theory represented as a set of predicates with assigned values. (d) An example of a logical expression defined over the predicates in (c). This expression corresponds to a VGDL syntax bias for "The sprites referenced in termination conditions should exist in the sprite set".

One side effect of using the predicate/value representation to form a theory space is that while every set of predicate/value pairs maps to a theory, not all of those theories are syntactically valid in VGDL. For example, the sprite types referenced in each termination condition must be defined in the sprite set in order for the VGDL theory to be executable. Changing the value of the predicate "SType(1)" from four to seven would result in a termination condition that references a non-existent seventh sprite type. While this theory can be encoded numerically, it violates VGDL syntax. Fortunately, the structure of the predicate/value representation makes it straightforward to define a logical statement that captures this syntax rule. Fig. 2d shows an example of such a statement. We can treat this logical statement as a bias function, which favors theories that match this syntax requirement. Following in this vein, we created a set of syntactic bias functions (described in detail in Appendix A) such that predicate/value pairs for which all syntactic biases $B_i(\theta) = 1$ are guaranteed to map to syntactically valid VGDL theories.

2.5. Defining semantic biases

To identify other biases that humans might have for learning video games, we turned to a database of existing games descriptions: the General Video Game AI (GVG-AI) competitive training sets (Perez-Liebana et al., 2016). GVG-AI is an annual competition for AI developers that is designed to encourage the creation of general game-playing algorithms. It contains a broad sample of games from different genres. In order to provide a consistent interface for training AI agents, the games have all been converted to VGDL descriptions. Most of these games are clones of popular games whose designs have been gradually honed over many generations of interactions between game players and designers. Iterated cultural transmission has been shown to produce stationary distributions that reflect people's prior beliefs (Kalish, Griffiths, & Lewandowsky, 2007; Tessler, Tsividis, Madeano, Harper, & Tenenbaum, 2021), suggesting that databases such as the GVG-AI game set are a useful avenue for exploring human inductive biases about games.

Searching through the GVG-AI database revealed several patterns that were indicative of plausible inductive biases (Table 1). Some of these patterns were specific to the domain of games. In particular, almost every game in the database had at least one winning condition and at least one losing condition, and those conditions tended to involve non-arbitrary goals. In other words, the goal of a game was almost always to collect/destroy/transform *all* of some kind of object rather than some arbitrary subset (e.g., collect 2 of the 5 coins). Taken together, these patterns suggest a bias that games should have at least one way to win and lose, and a bias that win/loss conditions should involve non-arbitrary goals.

We also found a set of patterns pertaining to the intended purpose of games. As the games in this data set were designed to be learned by humans, they tended to have structures that clearly communicated their intended use. For example, many objects in the GVG-AI games only serve a single function. If object A is a key that opens doors, it probably does not also serve as a bullet for a gun or a bridge for filling gaps. Also interactions were often informative about termination conditions: interactions that award points tended to provide information about how to win a game, while interactions that detracted points tended to provide information about how to lose a game. These patterns suggest biases for consistent object functions and informative point structures.

The GVG-AI games also tended to have specific causal structures. All the object kinds in a GVG-AI game were involved in moving, blocking, destroying, or otherwise interacting with at least one other kind of object. Furthermore, those interactions tended to move the player towards (or away from) at least one goal. In other words, all of the object kinds in a theory served at least one purpose. Similarly, goals and interactions were attainable in every GVG-AI game we explored. Here we use *attainability* to mean "at least one

Table 1
Summary of semantic biases incorporated into the prior $P(\theta)$.

Bias	High-level description	Formal description
Win/Lose	Games should have at least one way to win and one way to lose.	$[\ \exists\ g \in Goals$ s.t. $HasCond(g) \wedge Win(g) == True\ ] \wedge [\ \exists\ g \in Goals$ s.t. $HasCond(g) \wedge Win(g) == False\ ]$
Non-arbitrary Goals	Goals involve all of X rather than some of X.	$\forall g \in Goals \cdot Limit(g) == 0$
Attainability	For every goal/interaction, there is some combination of interactions that would allow the avatar to trigger that goal/interaction.	$[\forall i \in Ints.i \in ForwardChain(avatar)] \wedge [\forall g \in Goals.g \in ForwardChain[avatar]]$
Consistency	Objects should have consistent functions (e.g., if A destroys B, A should also destroy C)	$\forall i_1 \in Ints \cdot \forall i_2 \in Ints \cdot (i_1 == i_2) \vee Patient(i_1) \neq Patient(i_2) \vee Type(Effect(i_1)) == Type(Effect(i_2))$
Informativeness	Interactions that award positive points should be involved in triggering win conditions.	$\forall i \in Interactions \cdot (\ (Effect(i) == killWithScoreChange) => [\ (Value(i) > 0) => (\ \exists\ g \in Goals$ s.t. $Win(g) == True \wedge i \in RevChain(g)\ )\ ]$
Purpose	Every interaction/object kind should be involved in triggering at least one goal.	$[\ \forall i \in Ints \cdot \exists\ g \in Goals$ s.t. $i \in RevChain(g)\ ] \wedge [\ \forall k \in ObjKinds.\exists\ g \in Goals$ s.t. $k \in RevChains[g]\ ]$

sequence of interactions makes it possible for the preconditions of a target interaction or goal to occur". These patterns suggest a bias that all objects serve a purpose and that all goals are attainable. We implemented these notions of *purpose* and *attainability* by building a function that used knowledge of properties of VGDL to construct reverse and forward causal chains. Reverse causal chains described how goals were triggered by available interactions and how to trigger those interactions. If an interaction or object kind participated in at least one reverse chain, then we marked it as serving a purpose. In contrast, forward causal chains described which objects could affect other objects, and how those effects ultimately related to the goals of a theory. If an interaction or goal occurred in the forward causal chain beginning with the player-controlled avatar object, then we marked it as attainable. The algorithm that produced these causal chains is described in more detail in Appendix A.

The biases for consistency, informativeness, purpose, and attainability seem to reflect folk psychological tendencies to evaluate entities abstractly in terms of the purpose they were designed to achieve (Dennett, 1987). Games are often designed to teach you how to play, and providing informative points, attainable goals, and purposeful, consistent objects all helps in that regard. We refer to the six biases described in Table 1 as semantic biases.

2.6. Approximating the posterior

Now that we have defined a likelihood $P(\mathbf{D}|\theta)$ and prior $P(\theta)$, we can compute the posterior probability $P(\theta|\mathbf{D})$. To calculate this exactly, however, we would need to compute a normalizing constant so that the posterior probabilities sum to one. In practice, however, it is intractable to calculate the normalizing constant for a theory space of this size. Instead, we approximate the posterior using *discrete particle variational inference* (DPVI; Saeedi, Kulkarni, Mansinghka, & Gershman, 2017). The key idea is to replace the posterior distribution over the full set of possible theories with a much smaller set of high probability theories (particles). Each theory is assigned a weight, and the set of weights is optimized so that the particle approximation gets as close as possible to the true posterior.

Formally, the variational particle approximation consists of K theories $\{\theta_1, \ldots, \theta_K\}$ and associated weights $\{p_1, \ldots, p_K\}$, which together form a probability distribution over the theory space:

$$P(\theta|\mathbf{D}) \approx \sum_{k=1}^{K} \mathbb{1}[\theta = \theta_k] p_k, \tag{5}$$

where p_k is proportional to the unnormalized posterior probability of that theory, $P(\theta, \mathbf{D})$. A complete technical description of the algorithm can be found in Appendix B.

2.7. Using the model to estimate values

With a theory inference algorithm in hand, we return to the problem of using theories to estimate the value of sequential plans. Estimating a value function requires first specifying a reward function. The structure of VGDL theories provides a natural reward function. Many object interactions in a theory either directly award points or lead to completing termination conditions. These

termination conditions can also be given explicit positive or negative values to incorporate them into the reward function. In the current work we use values of 5 and −5 points for winning and losing respectively.

In our prior work on theory-based RL, we found that, in addition to external rewards, internal shaping rewards are also necessary for successful planning in complex tasks (Tsividis et al., 2021). We replicated the general structure of the reward function from that prior work here, with an additional bonus term that we will describe below:

$$R(s,a,s',\theta) = R_{ext}(s,a,s',\theta) + R_{int}(s,a,s',\theta) + R_{bonus}(s',a,P), \tag{6}$$

where $R_{ext}(s,a,s',\theta)$ represents external points earned after taking action a from state s and arriving in state s' (minus a constant cost c_{action} for every action taken), and the internal reward $R_{int}(s,a,s',\theta)$ is defined as:

$$R_{int}(s,a,s',\theta) = \sum_{g \in Goals(\theta)} \frac{|N_x(s') - N_x(s_0)|}{|Tar(x,g) - N_x(s_0)|}. \tag{7}$$

$Goals(\theta)$ represents the termination conditions in theory θ. All goals in the subset of VGDL used for the current work consisted of reaching a target amount of a certain object kind; thus the agent received an internal reward proportional to the remaining amount of each target object. $N_x(s')$ represents the number of objects of kind x in state s', and $Tar(x,g)$ represents the target count defined in g.

Given a reward function R, there are many ways to estimate the expected future reward for an action. Here we draw on an approach that we used previously (Pouncy et al., 2021), which involves identifying object-oriented actions, applying Monte Carlo Tree Search (MCTS) to generate internal simulations with the core knowledge engine, and then estimating an action-value function $Q(s,a,\theta)$ with these simulated trajectories by using the Sarsa(λ) algorithm (Rummery & Niranjan, 1994). The output of this algorithm, $Q(s,a,\theta)$, represents an estimate of the total future reward that an agent could expect to earn under theory θ by selecting the action a from state s and then following the optimal policy from that point on. Appendix A provides additional details about this algorithm.

In a classical model-based RL setting, the task model is known *a priori*. Here, however, we have uncertainty about the task model itself. Thus, rather than selecting actions to maximize future reward for a single theory, we instead want to identify actions that maximize reward after marginalizing over all possible theories (Martin, 1967). While it is computationally intractable to explicitly compute an action-value function in this case, there are a variety of ways of approximating this process. Here we use an approximation method proposed by Littman, Cassandra, and Kaelbling (1995) called Q_{MDP} that estimates theory independent Q-values $Q(s,a)$ by marginalizing theory-dependent q-values over a posterior, as shown below:

$$Q(s,a) = \int_\theta Q(s,a,\theta) P(\theta|\mathbf{D}) d\theta. \tag{8}$$

We can estimate this using our particle approximation of the posterior as follows:

$$Q(s,a) \approx \sum_{k=1}^{K} Q(s,a,\theta_k) p_k, \tag{9}$$

where θ_k and p_k are the theory and weight associated with the kth particle in our approximation, respectively.

One limitation of the Q_{MDP} approach is that it does not take into account the potential value of learning new information (i.e., reducing the uncertainty in the posterior over theories). This can lead to looping behaviors where the same actions are chosen over and over again due to incomplete information about the task theory (Littman et al., 1995). To get around this issue, we provided agents with an internal exploration bonus during the value estimation process. This bonus was designed to approximate the expected information gain for exploring a particular interaction, and was implemented as follows:

$$R_{bonus}(s',a,P) = \sum_{x_1,x_2 \in Unexplored(\theta)} C(s',a,x_1,x_2,P), \tag{10}$$

where $Unexplored(\theta)$ represents the set of interactions between object kinds x_1 and x_2 that the agent has not yet observed. $C(s',a,x_1,x_2)$ represents an internal reward for exploring the novel interaction between x_1 and x_2, defined below:

$$C(s',a,x_1,x_2,P) = \begin{cases} Z_H^{-1} H(x_1,x_2,P), & \text{if } a \mapsto (x_1,x_2) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $a \mapsto (x_1,x_2)$ means that action a induces an interaction between x_1 and x_2. $H(x_1,x_2,P)$ is the entropy of the outcome of the interaction between x_1 and x_2 under the posterior $P(\theta|\mathbf{D})$. This interaction entropy is easier to calculate than the expected uncertainty reduction over the full posterior, and provides at least an approximation of the informativeness of a given interaction. We calculated $H(x_1,x_2,P)$ using the posterior over theories:

$$H(x_1,x_2,P) = -\sum_e P(e|x_1,x_2) \log P(e|x_1,x_2) \tag{12}$$

where $P(e|x_1,x_2)$ is the probability of outcome e, marginalizing over θ. To ensure that the bonuses did not overwhelm the reward function in Eq. (6), we include a scaling factor Z_H which reflects the maximum potential entropy for a given posterior approximation. This maximum entropy occurs when every outcome for an interaction is equally likely. If the number of potential outcomes N_E is greater than the number of particles K, then the maximum entropy occurs when all outcomes are equally likely, $P_e = 1/N_E$, in which case $Z_H = \log N_E$.

Finally, agents used the Q-value function at every time step to selected the action with the highest estimated action value. To make planning more tractable, we also had agents cache and re-use their action-value estimates. Agents re-estimated action values every 4 steps, or whenever new rules were observed.

3. Experimental design

The framework described in the previous section sharpens our empirical questions. Does theory-based RL with semantic and syntactic biases better account for human learning behavior relative to theory-based RL with just the syntactic biases? To address this question, we evaluated human and artificial agent performance as they learned to play a series of video games. We also looked at human and agent predictions about their environment using a series of prompt questions that we designed to assay the biases of interest.

3.1. Participants

We recruited 27 participants to play a series of games using the CloudResearch interface to Amazon's Mechanical Turk platform (Litman, J. Robinson, & Abberbock, 2017). All participants were vetted using CloudResearch's standard attention and engagement measures to ensure high quality data[3]. Participants played three runs of seven different games. They received a base payment of $2.50 with a $0.20 bonus for each run that they won. Participants received an additional $0.20 bonus for each run where they earned the maximum points possible on that run. We removed 1 participant who won fewer than three of the 21 available game runs.

3.2. Instructions and tutorial

Upon beginning the task, participants were told that they would play three runs of several different games by using the arrow keys on their keyboard. They were then informed about the bonus structure of the task. Finally they were told that game rules might vary from game to game, and that they might be asked to answer some questions during the task. Participants then completed a short tutorial to familiarize themselves with the task structure.

During the tutorial, participants controlled a block and navigated a series of rooms. To get from the first room to the second room they had to collect an object that awarded one point. After witnessing this interaction, participant's attention was drawn to the right-hand side of the screen where a rule explaining that interaction (e.g., "green blocks award one point by destroying red blocks") was added to a list of observed rules. Participants were told that rules would be added to this box as they were discovered and then had to click a button to continue with the task. A few steps after collecting the first object, participants were shown an example of a question. An overlay popped up over the game and required them to answer two questions of the form "Given the rules you have observed so far, how likely is it that the following is also a rule of this game?" before they could continue playing the game. One question asked about the rule "when there are zero green blocks left you *win* the game", while the other question asked about the rule "when there are zero green blocks left you *lose* the game". Participants provided answers by sliding a scale from zero "very unlikely", to 100, "very likely". The order of these questions was randomized across participants, as were the colors of the objects in the game. Once both questions were answered, participants continued play, exploring other objects, observing other rules, and answering intermittent questions until they either won or lost the game. After one run of this tutorial game, participants were told that they were going to move on to the main task.

3.3. Main task

For the main task, participants played three runs of seven different games. Each game run involved the same object layout and rules. Between games participants were warned that they were switching to a different game. The object layout and rules would then change for the next game. To minimize the overlap between games, each game involved a variety of different kinds of interactions. The "Destroy the blocks" game focused on using one object to destroy multiple instances of another object, while the "Collect the key" game required using one object to pick up a key and use it to open a door. The "Fill the gaps" game involved transforming a set of intangible objects into blocks that could be used to remove dangerous gaps. The "A opens everything" and "A creates coins" games involved using the same object kind to solve multiple puzzles, while the "A pushes/B freezes" and "A transmutes/B destroys" games involved selecting from two objects with known functions to solve a novel puzzle. The names of these games were used only for data analysis and were not shown to the participants at any time. The layout and rules for each game are included in Appendix B. These games were presented in random order, and the color of the objects in each game was randomized across participants to minimize the effect of any associations with particular colors.

Just as in the tutorial, participants discovered rules by interacting with objects. These rules were added to the right-hand side of the screen as they were discovered. While we still highlighted the rule box with a red border to draw participant's attention when new rules were added, we no longer required them to click a box to continue for the main task. The observed rule list helped reduce the memory load caused by this task and control for any differences that might arise from giving our model the ability to learn game rules from a single observation. We also found that many participants enjoyed seeing new rules get added to the list and seemed to explore more as a result. While the structure of these internal rewards was not exactly captured in the model, it loosely paralleled the internal bonus that agents received for discovering informative rules. We will refer to the parts of the task involving selecting game actions to learn rules and maximize reward as the "planning sub-task".

[3] CloudResearch independently screens their participants with a series of attention checks, bot detection processes, and language comprehension tasks and we drew our participants from this pre-screened population.

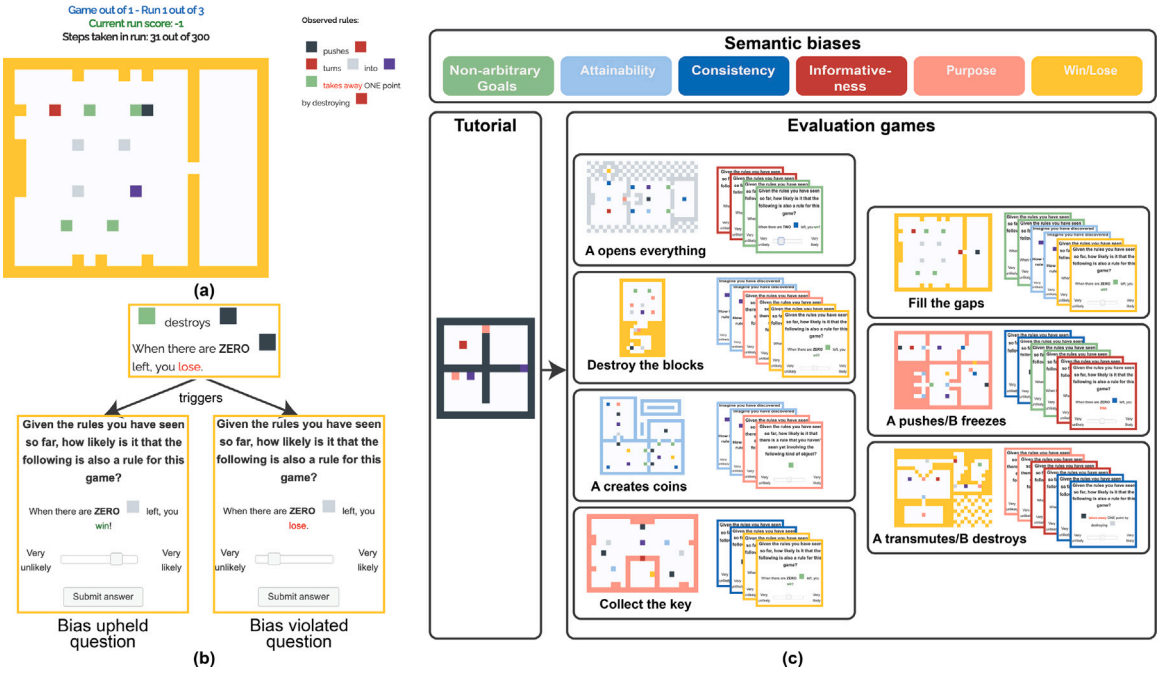


Fig. 3. Visualization of the experimental design. (a) Screenshot of a game as seen by participants. Previously observed rules were displayed to the right of the game. (b) An example trigger and question pair for the Win/Lose bias. When the participant triggered the "green kills black" lose condition, they were asked about the likelihood of another potential rule involving a win condition (the "bias upheld" question) and a lose condition (the "bias violated" question). (c) Experiment structure. Participants played one run of the tutorial, followed by three runs of each evaluation game. Participants were asked questions like those in (b) during each game. The colored images next to each game indicate the questions shown in that game. The color of the image border corresponds to the bias targeted by that question. The "Semantic biases" section provides a color key for the bias types. The color of the objects in the game screen were randomly chosen for each participant and do not reflect anything about the semantic biases represented in that game.

Following the tutorial, observing certain rules during the main task triggered pairs of questions. Questions were displayed a few steps after first observing the triggering rule to give participants time to notice the new rule before answering. Each pair of questions was designed to assay a particular semantic bias from Section 2.5. For example, shortly after participants first discovered that falling in a gap caused them to lose the "Fill the gaps" game, we presented them with the two questions shown in Fig. 3b. One question asked about the likelihood of an unobserved winning condition, while the other involved an unobserved second losing condition. As theories containing the former rule are guaranteed to uphold the win condition bias, while theories containing the latter rule could potentially violate the bias, we will refer to these questions as the "bias upheld" and "bias violated" questions, respectively. Participants who believe that games should have at least one win condition should evaluate the bias upheld rule as more likely than the bias violated rule. In contrast, participants without the win condition bias should evaluate both rules as equally likely.

Each question pair in the experiment was similarly constructed to contain one rule that would be more likely given a particular bias and one rule that would be less likely given that bias. Most of the questions were structured like the ones in the tutorial. Due to the nature of their associated biases, the purpose and attainability questions were structured differently. The purpose questions asked "Given the rules you have seen so far, how likely is it that there is at least one rule that you haven't seen yet involving the following kind of object?" The attainability questions asked "Imagine you have discovered the following rule: X. How likely would it be that the rule below was also a rule?" For example, an attainability bias question might ask "Imagined you have discovered the following rule: 'green objects transform blue objects into pink objects.' How likely would it be that 'you can push green objects' was also a rule?" The intuition behind this question framing is as follows: in order for the 'green objects transform blue objects into pink objects' rule to be attainable, there must be a way for green and blue objects to come into contact. Thus, if a theory contains this transformation rule, an agent with an attainability bias should infer that it also contains a rule that allows green and blue objects to interact, like 'you can push green objects.' In contrast the presence of the transformation rule should have no effect on the likelihood of an unrelated rule like 'you can destroy green objects.'

Participants answered three different question pairs for each of the six biases of interest. To ensure that response patterns for each bias were not a result of the specific structure of individual games, the three question pairs associated with each bias appeared in three different games. To avoid interrupting the flow of the game too much, we limited each game to only contain two or three question pairs. Fig. 3c provides a schematic of which biases we evaluated in each game. Appendix B provides a complete list of the question pairs associated with each bias along with the triggering interactions/games for each pair. All participants saw all 18 question pairs during a run. The question pairs appeared in the same games after the same interactions for all participants; due to the fact that participants played the games in random order and potentially explored games in different ways, each participant saw

the question pairs in a different order. The order of the questions within each pair was also randomized across participants. We will refer to the parts of this task involving predicting the answer to these question prompts as the "prediction sub-task".

4. Results

To address our central question of whether theory-based RL with semantic and syntactic biases better accounts for human learning behavior, we begin by looking at our prediction sub-task. First we demonstrate that the question pairs in this sub-task reliably assay the presence or absence of specific biases in our models. We then present evidence that human question responses match model predictions for three of the six semantic biases. Next we compare human and model behavior on the planning sub-task and find that while humans outperform both agents on this sub-task, agents with all semantic biases produce more human-like behavior than agents without them. In particular, agents with the semantic biases get stuck less and choose to target their exploration during learning in more human-like ways. We further show that while fitting bias weights to human prediction behavior results in more human-like results in the prediction sub-task, the same fitted weights produce less human-like behavior in our planning sub-task. We then use independent ratings of human and agent game play to look for differences in more complex, multi-step exploration patterns. We find that agents with semantic biases receive slightly higher ratings of directedness and competence, though our current implementation still fails to capture some broader nuances of human behavior. Taken together, these results suggest that human behavior is generally consistent with models that possess all of the semantic biases.

4.1. Model performance in prediction sub-task

To evaluate whether the question pairs in our prediction sub-task reliably assayed the presence or absence of their intended bias, we first simulated model behavior. We used two classes of theory-based agents for these simulations: agents with the syntactic biases prior, and agents with syntactic and semantic biases. We refer to the former as the syntactic agents and the latter as the semantic agents. Both agent types used the inference and planning mechanisms described in Section 2. The only difference between the two agents was in the set of bias weights $\boldsymbol{\omega}$ used in Eq. (4) for the prior $P(\theta)$. The syntactic agents had a fixed weight ω_s for all of the syntactic biases, and a weight of zero for all of the semantic biases. The semantic agents used the same fixed weight ω_s for the syntactic biases with a non-zero fixed weight ω_a for all six semantic biases. We fit ω_s and ω_a using a grid search over potential combinations of these parameters (using all combinations of integer values from -5 to 5). We found that setting these parameters to 4 and 3, respectively produced posteriors with the highest proportion of theories that matched all non-zero weighted biases for each agent type. We simulated model predictions by using the particle approximation of the posterior described in Eq. (5). Since each prediction question involved evaluating the likelihood of a rule r given prior observations $\mathbf{D}$, we estimated this conditional probability $P(r|\mathbf{D})$ by checking which of the K particles in an agent's posterior contained the rule r:

$$\begin{aligned} P(r|\mathbf{D}) &= \int_{\theta} \mathbb{1}[r \in \theta] P(\theta|\mathbf{D}) d\theta \\ &\approx \sum_{k=1}^{K} \mathbb{1}[r \in \theta_k] p_k \end{aligned} \tag{13}$$

To avoid the possibility that none of the K particles in a given posterior approximation contained the rule in question, we fixed half the particles in our variational approximation to ensure that they included the question rule. This allowed the model to approximate the posterior probability of specific rules more accurately. For additional details about the approximation algorithm, see Appendix A.

To provide a fair comparison to humans, we ran one instance of each agent type for every participant. We accounted for any differences in observations due to the order in which participants explored by yoking each agent to one human participant. Thus, for the prediction sub-task, each agent took the same actions as their corresponding participant before being evaluated on each question pair.

Fig. 4 shows the results of these simulations. The x-axis shows the six semantic biases from Section 2.5. The y-axis represents the average response difference between the bias upheld and bias violated questions for each of the three question pairs associated with each bias. The color of the bars represents the agent type, with blue representing the syntactic agents and orange representing the semantic agents. Green and red represent lesioned agents and fitted bias weights agents which we describe in detail below and in Section 4.4, respectively. The semantic agents consistently predicted that the rule in the bias upheld question was more likely than the rule in the bias-violated question for all six types of biases. In contrast, the syntactic agents predicted no difference. This makes sense intuitively as the syntactic prior only cares about the structure of rules in a theory, not their content. Thus, as long as both rules in a question pair were syntactically valid, both rules were equally likely. T-tests of the simulated response differences showed that the differences predicted by the semantic agent were significantly higher than those of the syntactic agent. Table 2 provides t-tests comparing the response differences in Fig. 4. We used Bonferroni corrections within each table column to correct for multiple comparisons.

We also looked at six lesioned versions of the semantic agents. Each of the six lesioned agents had the bias weight for one of the six semantic biases set to zero. By looking at the simulated predictions of the lesioned agents, we confirmed that the response differences on the question pairs for each bias type were largely independent. In other words, setting the informativeness bias weight to zero only affected the predicted response differences on the informativeness question pairs. Appendix C provides additional details about these analyses. The green bars in Fig. 4 show the performance of these lesioned agents. Here we only include the response differences that were relevant for each lesioned agent type. For example, the response differences for the lesioned agents on the

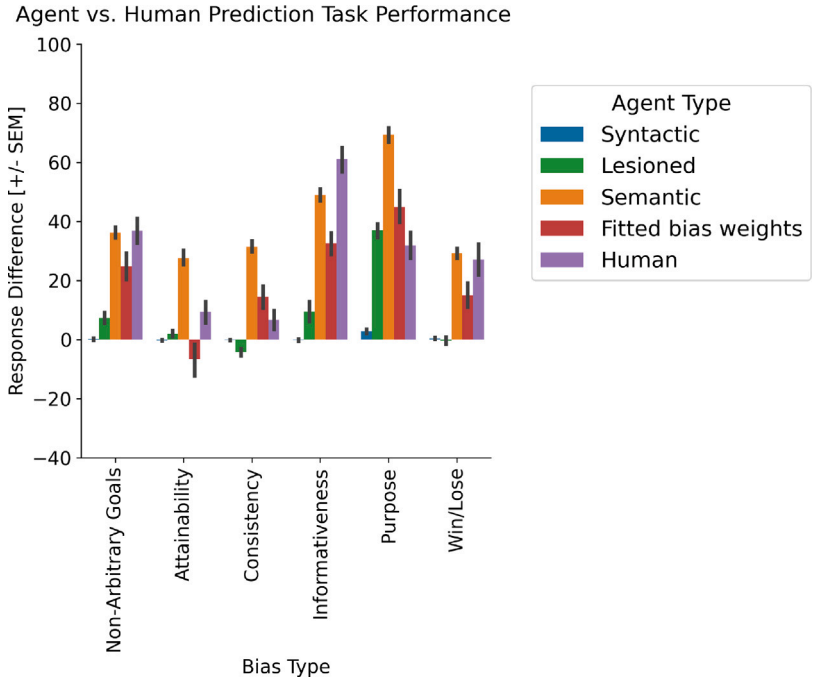


Fig. 4. Overview of human and agent performance on the prediction sub-task. *Y*-axis shows the average difference in response between the "bias upheld" and "bias violated" questions across all three question pairs for each semantic bias. Error bars reflect one standard error of the mean. The lesioned agent category only includes the agents with the associated bias removed (e.g., the green attainability bias type bar only includes agents with the attainability bias removed).

Table 2
P-values for t-tests comparing results from Fig. 4. Red text indicates cells where there was significant evidence of difference ($p < 0.05$). P-values were corrected for multiple comparison within each column using a Bonferroni correction.

Bias	Semantic vs. Syntactic	Fitted bias weights vs. Lesioned	Human vs. Lesioned	Human vs. Semantic	Human vs. Fitted bias weights
Non-arbitrary goals	**p<0.01**	**p<0.01**	**p<0.01**	p=1.00	p=0.33
Attainability	**p<0.01**	p=0.76	p=0.37	**p<0.01**	p=0.11
Consistency	**p<0.01**	**p<0.01**	**p=0.01**	**p<0.01**	p=0.76
Informativeness	**p<0.01**	**p<0.01**	**p<0.01**	p=0.05	**p<0.01**
Purpose	**p<0.01**	p=0.96	p=1.00	**p<0.01**	p=0.32
Win/Lose	**p<0.01**	**p<0.01**	**p<0.01**	p=1.00	p=0.43

consistency bias questions only reflect the lesioned agents where the consistency bias weight was set to zero. Similarly, the differences for the lesioned agents on the win/lose bias only reflect the lesioned agents where the win/lose bias was set to zero. The agents with lesioned biases for purpose, informativeness, and non-arbitrary goals showed significantly higher differences than the syntactic agent, suggesting that some of the other biases played some role in answering these questions. However, all lesioned agents showed significantly lower performance than the regular semantic agents, indicating that our question pairs at least predominantly relate to a single bias. Taken together, these results provide evidence that the questions in our prediction task can be used to reliably assay the presence or absence of particular biases as formulated in our model.

4.2. Evidence for biases in human prediction behavior

Human behavior in the prediction sub-task is consistent with the presence of some, but not all of the semantic biases that we explored. The purple bar in Fig. 4 shows human performance across all question pairs. Humans produce responses that are substantially higher than the lesioned agents for the question pairs associated with four of the semantic biases (non-arbitrary goals, win/lose, consistency and informativeness) but not for the attainability or purpose question pairs. However, while there was no significant difference between human and semantic agents on the non-arbitrary, win/lose and informativeness biases, human responses for consistency bias question pairs were significantly lower than the responses predicted by the semantic agents. Thus

human behavior in the prediction task is consistent with semantic biases for non-arbitrary goals, winning/losing conditions, and informative point structures, but not with semantic biases for attainability, consistency, or purpose.

4.3. Comparing human and agent performance on the planning sub-task

To further explore whether humans exhibit the semantic biases from Section 2.5, we also compared human and agent behavior in the planning sub-task. Just like with the prediction sub-task we ran 27 instances of the syntactic and semantic agents, one for every human participant. However, for the planning analysis, we allowed the agents to select their own actions rather than yoking them to a human participant. Letting agents select their own actions sometimes resulted in agents pushing necessary objects into corners or destroying them. Once this happened agents were effectively stuck, leading them to select actions at random. To address this, we introduced a step limit and automatically ended games when an agent reached 250 steps. Since the maximum number of steps in each game was much larger than the average number of steps required to win or lose, agents only produced this third outcome when they got stuck.

On average, human participants won game runs more often than either kind of agent. Fig. 5a shows the percentage of runs that ended in a win on the y-axis and agent type on the x-axis. There was no significant difference in win percentage between the syntactic and semantic agents. However, task performance alone is not always the best measure of human-like behavior. For example, RL agents often exhibit patterns of repeated interactions that humans do not. To explore this, we drew on the underlying structure of the seven evaluation games to group object interactions into a few broad categories. All of these games essentially involved using push-able tool objects to either trigger or avoid interactions with non-mobile target objects. While tools could be pushed into each other, these tool/tool interactions rarely had any effect. Often there were also rewarding objects to collect, obstacles that blocked progress, and punishing objects that killed the player. Fig. 5b shows the proportion of interactions belonging to each of these types that agents explored during learning, averaged across all seven games. Since agents and humans were provided with rules as soon as they were observed, we classified learning interactions as all actions taken up to and including an agent's first time triggering the win condition for a particular game.

For all interaction types except the Avatar/Blocker type, the semantic agents exhibited exploration preferences that were significantly closer to human preferences than those of the syntactic agents. In particular, agents with the semantic biases spent less time pushing tool objects into walls or other tools and more time using the tool objects to interact with target objects. This resulted in them achieving a higher proportion of interactions with rewarding objects and getting stuck less. Fig. 5c shows the proportion of runs during learning where the agent ended the game by running out of steps. All agents showed similar patterns of behavior in runs where they got stuck. On runs where agents did not get stuck, the syntactic agents explored significantly more tool/tool interactions ($t(359) = 2.36$, $p < 0.05$) while semantic agents spent more time exploring tool/target interactions ($t(359) = 2.55$, $p < 0.05$). This more directed exploration helped the semantic agents get to final part of each game more often, while the syntactic agents often got stuck in the earlier parts of each game. However, when presented with the chance to trigger the winning and losing conditions in the later parts of each game, the semantic biases were insufficient to push the semantic agents towards the winning interactions. Thus the semantic agents did not win as often as humans, even though they got stuck less.

4.4. Fitting biases based on human prediction behavior

Human performance on the prediction sub-task was consistent with some but not all of the semantic biases. However, the agents with all the semantic biases produced more human-like behavior on the planning sub-task. To further understand this combination of results, we also evaluated whether agents with only the set of biases exhibited by humans in the prediction sub-task still produced human-like behaviors in the planning sub-task. We accomplished this by evaluating a series of fitted bias agents. For these fitted bias agents, we used the same ω_s of 4 for all syntactic biases, but fit the semantic bias weights separately for each participant. Attempting to fit these bias weights using classical model fitting approaches proved to be computationally intractable, so instead we used a confirmatory factor analysis (CFA) to estimate the latent bias weights from performance on the prediction task. The central assumption behind a CFA is that a series of response variables $\mathbf{y}$ are linearly related to a set of latent factors $\boldsymbol{\eta}$. For the current work, $\mathbf{y}$ corresponds to the set of response differences on the prediction tasks, while the factor scores $\boldsymbol{\eta}$ are linearly related to the semantic bias weights $\boldsymbol{\omega}$ for each participant. In order to properly re-scale the $\boldsymbol{\eta}$ factor scores into the same space as the bias weights $\boldsymbol{\omega}$, we included the semantic and lesioned agent responses along with the human participant responses when fitting the CFA. We could then use the mean factor scores for the lesioned and semantic agents (whose bias weights were known) to re-scale the factor scores for the human participants into the bias weight space. This approach naturally takes into account any difference in response variance between the different prediction questions associated with each bias.

After evaluating several different CFA models (described in more detail in Appendix D), our final model had a χ^2 of 378.6 with 125 degrees of freedom and a RMSEA of 0.091 (90% confidence interval: (0.081, 0.102)). Under the CFA fit rules proposed by Bentler and Hu (1999), this model is right on the border between fair and poor. However, these metrics predominantly measure the model's ability to account for question responses on this task, which were inherently noisy. We also looked at the model's ability to capture differences in the known bias weights from the agents, where the CFA reliably estimated higher weights for the semantic than the lesioned agents. Furthermore, we found that the question responses predicted by agents with the fitted bias weights did significantly correlate with human responses for four out of the six biases studied[4], and Fig. 4 shows that the agents with the fitted bias weights

[4] See Appendix D for more details about the fitted bias weights and their correlation patterns.

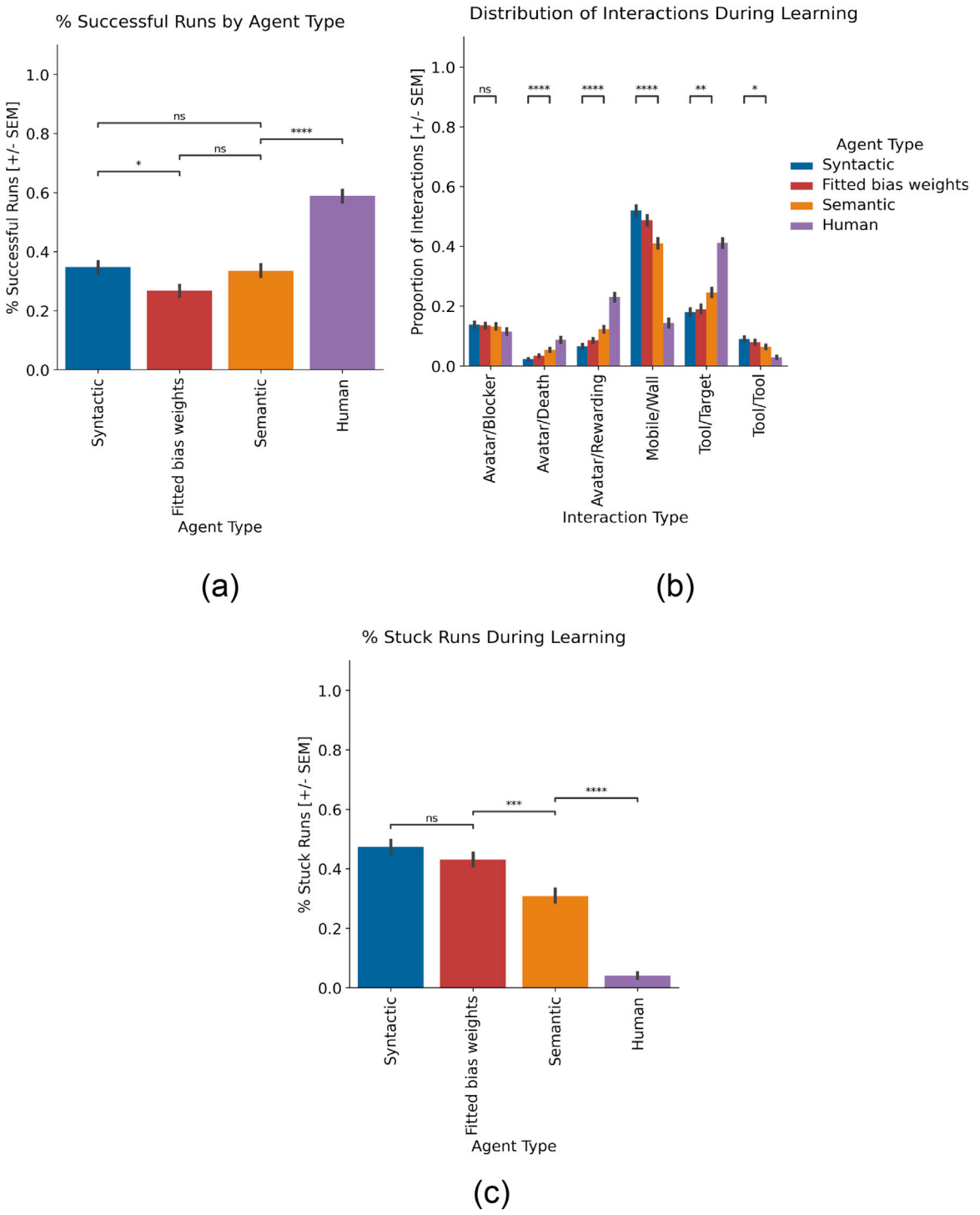


Fig. 5. Overview of human and agent performance on the planning sub-task. (a) shows the total percentage of runs where agents were successful. (b) show the proportion of different kinds of interactions triggered by an agent during learning. P-values represent a comparison between syntactic and semantic agents with Bonferroni correction for multiple comparisons. (c) shows the percentage of game runs where agents got stuck during learning. Error bars show the standard error of the mean.

provided a more qualitative match to human predictions overall. In particular, the fitted bias agents better match the pattern of large and small prediction differences demonstrated by humans across bias questions, although the semantic agent produced differences that were closer to human responses on three out of the six question types. This suggests that while our approach was imperfect, it was adequate for our current purposes.

Fig. 5 shows that the behavior of the fitted bias agents in the planning sub-task was more similar overall to the syntactic agents than the semantic agents. Furthermore, there was no significant correlation across participants between any of the fitted bias weights and success rate, rate of getting stuck, or proportion of interaction type in the planning sub-task. Similarly, there was no correlation between the success rate, rate of getting stuck, or proportion of interaction type for human participants and the agents with that participant's fitted weights. Overall these results suggest that the attainability, purpose, and consistency biases that did not account for human question responses were necessary for producing more human-like behavior in the planning sub-task. Given that the attainability and purpose questions were the only two sets of questions that were differently phrased, it is possible that this different phrasing introduced a difference between how humans interpreted the questions and how we evaluated the questions using the model. Alternatively it is possible that humans possess semantic biases that we have not modeled here that affected either their prediction or planning behavior. Further research is needed to properly disambiguate these results.

4.5. Rating artificial agent behaviors

While looking at proportions of interaction types was useful for detecting some of the most blatant differences in human and agent behavior, we also observed broader differences in the gestalt character of agent planning behavior that were not well accounted for with these descriptive statistics. More than any individual choice in any individual, much of the "humanness" of behavior in these kinds of complex tasks comes across in more abstract behavioral properties like the "directedness" of human approaches. For example, humans and semantic agents who discovered that the tool objects in the "Fill the Gap" game (shown in Fig. 2) could be used to transform target objects often re-used that tool to immediately transform all of the other target objects on the screen. Syntactic and fitted bias agents, on the other hand, rarely exhibited this streamlined pattern of behavior. Understanding the source of these differences in gestalt behavioral patterns is an important part of understanding the similarities and differences between human intelligence and our current best models of cognition. To explore whether these gestalt behavioral patterns are an emergent property of inductive biases in decision making, we identified a few such characteristics and had independent human raters evaluate the presence or absence of these characteristics in videos of both human and agent performance on our planning sub-task. Initial exploration during a more open-ended pilot study suggested that raters found human play to be more directed and competent, so we chose "directedness", and "competence" as gestalt qualities to evaluate. We also included a more general "humanness" characteristic to capture other high level patterns of behavior not included in our other two characteristics.

We had a separate set of 87 participants familiarize themselves with each game and then compare videos of the 27 original humans and agents learning to play that game. After playing each game, raters watched a video of a human, semantic agent, and syntactic agent playing the same game with color-matched objects. Given the length of the rating task and the similarity of the fitted bias and syntactic agents on the planning task, we chose not to include the fitted bias agents in the rating task. After each video the raters were asked "Was the agent in this video a human or a computer?", "How competent was the agent you just watched?", and "How purposeful were the agent's actions?". Participants answered each question using a sliding scale from 0 ("Definitely a computer", "Not competent", or "Not purposeful/random") to 100 ("Definitely a human", "Very competent", or "Very purposeful"). The order of the games were randomized across participants and the order of the agent types was randomized across trials. Appendix E provides additional details about the rating task and participants.

Overall, raters were capable of distinguishing the three kinds of agents based on behavior. Because individual raters varied in the proportion of the rating bar they used, we instead focused our analyses on the within-participant rating difference. Fig. 6a shows rating differences between semantic and syntactic agents and humans and syntactic agents. Humans were rated as a more competent ($t(1012) = 12.43$, $p < 0.001$) and directed (($t(1012) = 11.83$, $p < 0.001$) than syntactic agents. Semantic agents were also rated as more competent (($t(1012) = 2.61$, $p < 0.01$) and directed (($t(1012) = 2.63$, $p < 0.01$) than syntactic agents, though to a much lesser degree. Competence and directedness ratings within participants were also strongly correlated.

There was no significant difference in the average humanness ratings for any of the three agents. On closer inspection, however, this seems to be a result of how raters interpreted what it means for an agent to be human-like. Some of the raters consistently rated *more* competent agents as more human-like, while others rated *less* competent agents as more human-like (see Appendix E for correlation analysis). This suggests that some raters expected humans to be better than computers at these kinds of games, while others thought the opposite. Fig. 6b shows that when we split the raters into these "more human = more competent" and "more human = less competent" groups based on individual correlation between competence and humanness ratings, we found that the former group rated human agents as significantly more human ($t(274) = 6.08$, $p < 0.01$) than syntactic agents while the latter rated humans as significantly less human ($t(318) = 4.85$, $p < 0.01$) than the syntactic agents. The former group also rated the semantic agents as significantly more human ($t(274) = 2.01$, $p < 0.05$). There was no significant difference between the semantic and syntactic agents for the "more human = less competent" group.

To better understand the source of these rating differences, we also fit a mixed effects model predicting each rating as a function of the success rate, rate of getting stuck, proportion of interaction types, and agent type. The coefficients for success rate, rate of getting stuck, and proportion of interactions with walls were significant for all three ratings. This suggests that the rating differences between semantic and syntactic agents primarily capture the same pattern observed in the planning behavior: semantic agents and

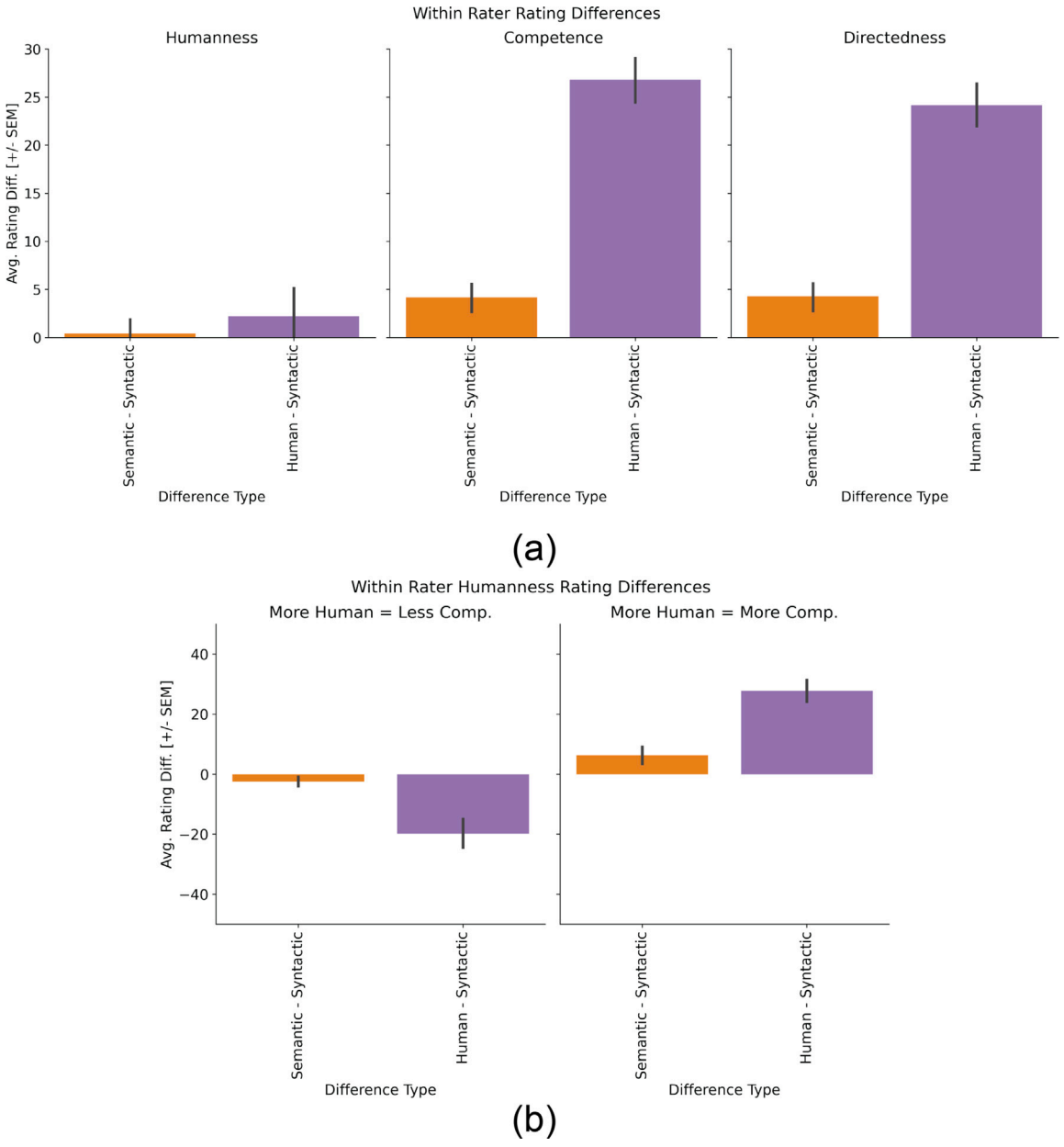


Fig. 6. (A) Gestalt behavioral characteristic ratings for human and agent planning behavior. Each graph shows the within-rater difference scores between ratings of the semantic and human agents relative to the syntactic agents. The *Y*-axis in the left-most figure represents the rater's certainty that video was of a human agent. Middle figure shows participant's rating of agent competence. Right-most figure shows participant's rating of the purposeful-ness of an agent's actions. Differences in competence and directedness ratings were significant at the $p < 0.05$ level. (b) Humanness ratings split between raters who believed more competent agents were *more* likely to be humans and raters who believed more competent agents were *less* likely to be human. Error bars show standard error of the mean.

humans avoid pushing objects into walls, use tools correctly, and avoid getting stuck in tasks. This makes them appear more directed and competent.

Agent type for the mixed effects model was encoded with dummy variables for humans and semantic agents. The coefficient for the semantic agent variable was not significant, suggesting that the semantic agent did not demonstrate any more nuanced patterns of behavior that reliably distinguished them from the syntactic agents. In contrast, the coefficient for the human variable was significant, suggesting that humans behavior did vary in some way not captured by the other metrics that we used to analyze planning behavior. Appendix E provides a more thorough description of this analysis.

Taken together these results suggest that the semantic biases lead to differences in the gestalt character of planning behavior that were strong enough to be observable to untrained human raters. However, the semantic biases we have explored so far were

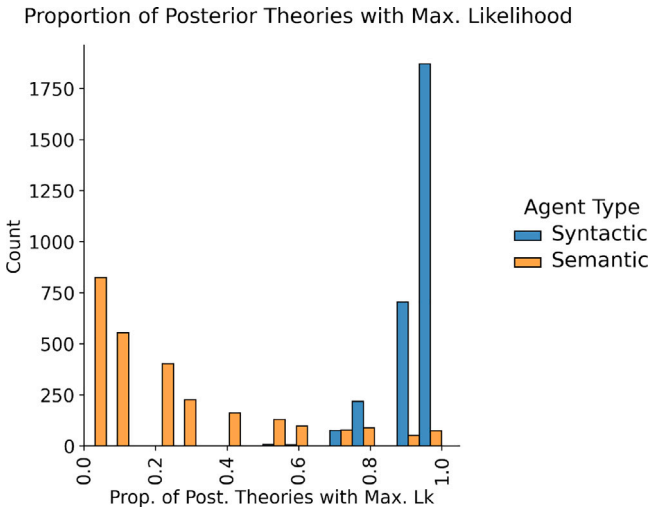


Fig. 7. An overview of the effectiveness of our DPVI algorithm for approximating posteriors during the planning sub-task. During planning, agents frequently re-estimated a posterior over theories $P(\theta|\mathbf{D})$. The x-axis represents the proportion of theories in each of these estimated posteriors that had the maximum likelihood score (i.e., satisfied all of the biases in the prior $P(\theta)$ that had non-zero weights). The y-axis represents the number of estimated posteriors with that proportion. Color indicates agent type. Blue represents the syntactic agents, while orange represents the semantic agents.

not sufficient to capture the full range of gestalt behaviors exhibited by humans in our task. Understanding nature of these patterns and their cognitive origins is an important direction for future research.

4.6. Effects of posterior approximation on agent behavior

While the semantic agents produced more human-like behavior, they still fell short of human performance in the planning sub-task. There are many potential explanations for this difference; here we highlight one explanation here for further analysis. One reason for superior human performance is that humans may make use of better approximate inference algorithms than the ones we used here. Upon closer inspection, the DPVI technique that we used often under-estimated high likelihood areas of the posterior for the semantic agents specifically. Fig. 7 shows the proportion of theories in an agent's approximate posterior that satisfied all the non-zero weighted bias functions. For the syntactic agents, most of the theories in the approximate posterior satisfied all of the syntactic bias constraints. However, the same process often failed to produce even a single particle that satisfied all of the inductive bias constraints for the semantic agents. And yet in many of these cases there are dozens, if not hundreds, of valid VGDL theories capable of explaining an agent's observations which match all the semantic constraints.

The asymmetrical performance of the DPVI algorithm under the syntactic and semantic bias weight sets is likely a result of the way we implemented the algorithm with our theory representation. As mentioned in Section 2, DPVI operates by weighting particles based on their unnormalized posterior probability. We constructed the particle set by searching within a neighborhood around a set of initial particles. Each neighbor theory was constructed by changing one of the predicate/value pairs associated with a theory to any of the other values that a predicate could take on. This resulted in very small differences between theories and their neighbors. This small step size was fine for the syntactic biases. Most of the syntactic biases have to do with individual lines of a theory, so they can be fixed with a few of these small steps. However, most of the semantic biases involved more complex links between multiple lines of a theory. A theory that violates the purpose bias, for example, may require dozens of small changes to multiple lines to produce a valid theory. Furthermore, many of these small intermediate changes may temporarily violate other biases. Intuitively, this suggests that adding the semantic biases made the DPVI algorithm more likely to get stuck in local optima rather than exploring the theory space more broadly. Including theories that are a little further away in the set of neighbors might help improve performance, bringing our model closer to human-level performance.

5. General discussion

What kinds of inductive biases do humans bring to complex tasks? We sought to answer this question in one complex task domain (grid-based video games) which incorporates elements of many other, more realistic domains: object-oriented, relational structure combined with events and goals. We developed an experimental approach to measuring and modeling inductive biases in this domain, and used this approach to evaluate whether model-based RL with syntactic biases for theory-based representations was sufficient to account for a variety of human-like learning behaviors.

In particular, we designed a series of video game environments and had both human and artificial agents complete two sub-tasks in these environments. First we let agents freely explore the environment, planning out actions that would allow them to learn

Table 3
Summary of VGDL predicates used in this work and their associated return types.

Predicate	Return type	Description
HasSpriteType(**SpriteType**)	**Boolean**	Indicates whether the *n*th line of the SpriteSet is filled in or not.
SpriteFeature(**SpriteType**)	**TypeFeature**	Indicates the functional type of the *n*th sprite type.
SpriteColor(**SpriteType**)	**Color**	Indicates the color of the *n*th sprite type.
HasInteraction(**Interaction**)	**Boolean**	Indicates whether the *n*th line of the InteractionSet is filled in or not.
InteractionAgent(**Interaction**)	**SpriteType**	Indicates which sprite type is the agent for the *n*th interaction.
InteractionPatient(**Interaction**)	**SpriteType**	Indicates which sprite type is the patient for the *n*th interaction.
InteractionEffect(**Interaction**)	**Effect**	Indicates the effect of the *n*th interaction.
HasParameter(**Interaction**, **ParamType**)	**Boolean**	Indicates whether the *n*th interaction includes a value for a particular type of parameter.
ParamValue(**Interaction**, **ParamType**)	Either **Integer**, **PositiveInteger**, or **SpriteType** depending on param type	Indicates the specific value for a given parameter type for the *n*th interaction.
HasCondition(**TerminationCondition**)	**Boolean**	Indicates whether the *n*th line of the TerminationSet is filled in or not.
LimitParam(**TerminationCondition**)	**PositiveInteger**	Indicates the limit associated with the *n*th termination condition.
WinParam(**TerminationCondition**)	**Boolean**	Indicates whether the *n*th termination condition is a winning condition or a losing condition.
STypeParam(**TerminationCondition**)	**SpriteType**	Indicates the sprite type associated with the *n*th termination condition.

environment dynamics and maximize reward. This free exploration then triggered intermittent question prompts asking the agents to predict the likelihood of novel rules based on their current understanding of the environment. These question prompts were divided into categories, with each category being designed to assay the presence/absence of a particular inductive bias about the semantic content of theory-based representations. We then compared human performance on these planning and prediction sub-tasks to the performance of theory-based RL agents with and without these semantic inductive biases. Finally, we explored whether these same semantic biases might account for more general gestalt properties of human planning behavior by using independent human raters to evaluate the directedness, competence, and humanness of human and artificial agent gameplay videos.

Overall, human planning behavior was better accounted for with agents possessing both syntactic and semantic biases. In particular, we found evidence for the following semantic biases in human planning behavior:

1. Games have at least one winning condition and at least one losing condition.
2. Goals are non-arbitrary (the goal of a game is to collect/destroy/transform *all* of some kind of object).
3. Points are informative (interactions that award points usually provide information about how to win a game, while interactions that detract points usually provide information about how to lose a game).
4. Objects serve consistent functions.
5. All target interactions or goals are attainable through at least one sequence of interactions.
6. All objects serve a purpose (i.e., they are involved in moving, blocking, destroying, or otherwise interacting with at least one other kind of object).

Human responses in the prediction sub-task were consistent with three of the six semantic biases (win/lose, non-arbitrary goals, and informative of reward structures). The lack of evidence for the remaining three biases may be a result of the particular framing of our question prompts. Additionally, agents with all six semantic biases were rated as demonstrating more human-like patterns of gestalt behavior. Taken together these results suggest that human learning in complex task domains involves not just inductive biases about the syntactic structure of learned representations, but also biases about their semantic content.

Our findings are broadly consistent with prior work on human biases in several other domains. For example, existing work has shown that humans develop domain-specific preferences for particular solution structures (Chi et al., 1981; Schoenfeld & Herman, 1982; Weiser & Shertz, 1983). The biases we found that favor classically structured games (i.e., at least one win/lose

Table 4
Summary of type value ranges for each variable type.

Variable type	Potential values
SpriteType	1–10
TypeFeature	MovingAvatar, Passive
Color	red, blue, green, yellow, black, gray, light gray, dark gray, purple, orange, light blue, gold, pink.
Interaction	1–12
Effect	killSprite, stepBack, bounceForward, transformTo, killIfOtherHasMore, collectResource, killIfHasLess, killBoth, killAll, killWithScoreChange, nothing
TerminationCondition	1–3
ParamValue	value, resource, limit, stype
Integer	−1, 0, 1
PositiveInteger	0–5
Boolean	True, False

condition and conditions involving non-arbitrary-goals) could be interpreted as similar domain-specific preferences in theory learning. Additionally, previous work has highlighted how beliefs about designer intent bias human category learning (Kelemen & Carey, 2006) and solution search (Duncker, 1945), and the biases we find for informativeness, consistency, attainability, and purpose could be interpreted as evidence for a similar design stance in theory learning. This broader literature suggests that similar biases apply to a wide swath of decision making domains, not just video games specifically.

We in no way intend to suggest that the six semantic biases studied here exhaust the space of human inductive biases in complex tasks. To this end, the current work is also intended to provide a novel computational framework for identifying other biases in these kinds of tasks. Using Markov logic networks as a prior allows for the integration of any set of conceptual biases that can be encoded as logical expressions over theory predicates. Defining biases over theory-based representations makes it easier to encode high-level biases that would be difficult to formalize over lower-level task representations. Furthermore, the weighted format of the prior makes it easy to empirically evaluate the effects of individual biases. We therefore also present the current work as a proof-of-concept for studying other biases in this domain.

Furthermore, while our current work has focused on biases in video game learning specifically, our general framework could also be used to explore biases in other complex learning domains. Theory-based RL is agnostic to the domain language used to encode theories, so our framework could be similarly applied to other domains by modifying this language. For example, while we did not include these components here, VGDL is capable of modeling many different kinds of agents. These agents can incorporate notions of belief, desire, and intent, and our framework could begin to include biases relating to these psychological notions as well. Similarly, with more robust core knowledge engines, the same approach could explore biases in a wide range of real-world decision making tasks. Thus, the current work provides not just an analysis of biases in human video game play, but also a more general approach for studying biases in real-world theory learning.

5.1. Limitations and future directions

While biases in theory learning play an important role in explaining human behavior, they are not the only piece of the puzzle of human intelligence. Even with the semantic biases, our agents still don't present a full account of human learning. We have already discussed how our choice of theory inference algorithm struggles with more complex biases. Similarly, our choice of value estimation process may not capture key elements of human cognition. For example, in spite of their successes, semantic agents still get stuck more than humans. One possible explanation for this is that humans also have inductive biases about what successful strategies look like for a task (cf. Rosman & Ramamoorthy, 2012; Wingate, Diuk, O'Donnell, Tenenbaum, & Gershman, 2013; Wingate, Goodman, Roy, Kaelbling, & Tenenbaum, 2011). Prior experience may have led humans to infer that pushing things into walls rarely helps, thus humans do not even consider these actions when evaluating plans. Identifying human inductive biases over action policies represents an interesting direction for future research.

Our work also raises several other interesting questions for future work. First, where do these biases come from? Children readily learn structured theories of the world from a young age (Gopnik et al., 2004), but it is not clear whether they do so using the same combination of inductive biases that adults exhibit. Second, could these kinds of biases be learned from experience? The particular biases that we have presented in this work can all be specified in predicate logic, and could potentially be learned through probabilistic program induction (Liu et al., 2019; Pu, Ellis, Kryven, Tenenbaum, & Solar-Lezama, 2020). It would be interesting to explore whether program induction could be used to learn inductive biases directly from experience. Third, are these biases mutable? Humans demonstrate contextual adaptation to signals like overall reward frequency (Gershman & Niv, 2015), so it is plausible that repeated exposure to counter-intuitive games (e.g., games with purpose-less objects or uninformative reward structures) could eventually lead participants to learn context-specific exceptions to particular biases.

Table 5
Summary of syntactic inductive biases. For simplicity, the Formal Description column only shows partial expressions where the rest of the expression can be inferred from the high level description.

Syntactic bias	High-level description	Formal description
Avatar exists	At least one sprite can be controlled by the player.	$\exists s \in$ **SpriteType** s.t. $HasSpriteType(s) \wedge SpriteFeature(s) ==$ MovingAvatar
InteractionSet non-empty	VGDL description includes at least one interaction rule.	$\exists i \in$ **Interaction** s.t. $HasInteraction(i)$
Agent/Patient exist	The agent, patient, stype, and resource sprite types are defined in the SpriteSet for all interactions	$\forall i \in$ **Interaction** $\cdot HasInteraction(i) \Rightarrow HasSpriteType(InteractionAgent(i))$
Unique agent/patient set	A VGDL description only describes one effect for every set of agent/patient sprite types.	$\forall [i_1, i_2] \in$ **Interactions** $\neg HasInt(i_1) \vee \neg HasInt(i_2) \vee [\{IntAg(i_1), IntPat(i_1)\} == \{IntAg(i_2), IntPat(i_2)\} \Rightarrow IntEff(i_1) == IntEff(i_2) \vee stepBack \in set(IntEff(i_1), IntEff(i_2))]$
Effects have correct parameters	killAll and transformTo effects have stype parameters, killIfOtherHasMore and killIfHasLess effects have resource and limit parameters, collectResource effects have limit and value parameters	$\forall i_1 \in$ **Interaction** $[HasInt(i_1) \wedge HasIntPar(i_1, SType)] \Leftrightarrow [HasInt(i_1) \wedge IntEff(i_1) \in transformTo, killAll]$
TerminationSet non-empty	VGDL description includes at least one goal.	$\exists g \in$ **TerminationCondition** s.t. $HasCondition(g)$
TerminationSet sprites exist	Goal sprites defined in the sprite set.	$\forall g \in$ **TerminationCondition** $\cdot HasCondition(g) \Rightarrow HasSpriteType(ConditionParam(g, SType))$
Unique sprite/limit set	Each goal has a unique sprite and limit combination. Thus the same condition cannot simultaneously trigger winning and losing.	$\forall [g_1, g_2] \in$ **TerminationCondition** $\cdot \neg HasCond(g_1) \vee \neg HasCond(g_2) \vee [CParam(g_1, Lim) == CParam(g_2, Lim) \wedge CParam(g_1, SType) == CParam(g_2, SType) \Leftrightarrow g_1 == g_2]$

Table 6
Parameter values for theory inference.

Parameter	High-level description	Value
N_A	Number of variational distributions to approximate	10
K	Number of particles for each variational distribution	10
N_S	Number of vector components to sample	50
ϵ	Variational free energy threshold	0.0001
ω_s	Base weight for all syntactic biases	4
ω_d	Base weight for all semantic biases	3
w_S	Likelihood weight for correct sprite set predictions.	1
w_I	Likelihood weight for correct interaction set predictions.	10
w_G	Likelihood weight for correct termination set predictions.	20

5.2. Conclusion

In conclusion, our work takes a step towards explaining how humans employ structured inductive biases in the service of learning complex tasks. It joins other recent work emphasizing the importance of inductive biases for explaining human performance and emulating it in artificial agents (Gershman, 2021; Griffiths et al., 2010; Lake et al., 2017; Sinz, Pitkow, Reimer, Bethge, & Tolias, 2019). The inductive biases we studied may partly explain the remarkable sample efficiency and flexibility of human learning.

Acknowledgments

This research was supported by grants from the Office of Naval Research (N00014-17-1-2984), the Multi-University Research Initiative Grant (ONR/DoD N00014-17-1-2961) and the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. We are grateful to Fiery Cushman, Josh Greene, Patrick Mair, Josh Tenenbaum, Pedro Tsividis, and Tomer Ullman for insightful discussions, Akshara Methukupalli for her coding help, and to Momchil Tomov for comments on an earlier draft.

Table 7
Full list of hierarchical movement options for detecting object-oriented actions. Bold text indicates hierarchical option names. Italic text indicates non-hierarchical option names.

Option name	Alternative number	Sequence
MoveTo	1	**AlignX**, **AlignY**
	2	**AlignY**, **AlignX**
PushTo	1	**MoveToPushObj**, *RotateIntoPlaceY*, **PushAlignY**, *RotateIntoPlaceX*, **PushAlignX**
	2	**MoveToPushObj**, *RotateIntoPlaceX*, **PushAlignX**, *RotateIntoPlaceY*, **PushAlignY**
AlignX	1	*MoveUntilAlignedX*
	2	*MoveUntilBlockedX*, *MoveUntilClearX*, *SingleStepX*, *MoveUntilClearY*, **MoveTo**
AlignY	1	*MoveUntilAlignedY*
	2	*MoveUntilBlockedY*, *MoveUntilClearY*, *SingleStepY*, *MoveUntilClearX*, **MoveTo**
MoveToPushX	1	*MoveUntilTouchX*
	2	*MoveUntilBlockedX*, *MoveUntilClearX*, *SingleStepX*, *MoveUntilClearY*, **MoveToPush**
MoveToPushY	1	*MoveUntilTouchY*
	2	*MoveUntilBlockedY*, *MoveUntilClearY*, *SingleStepY*, *MoveUntilClearX*, **MoveToPush**
PushAlignX	1	*PushUntilAlignedX*
	2	*PushUntilBlockedX*, *RotateIntoPlaceY*, *PushUntilClearX*, *RotateIntoPlaceX*, *SinglePushX*, *PushUntilClearY*, **PushTo**
PushAlignY	1	*PushUntilAlignedY*
	2	*PushUntilBlockedY*, *RotateIntoPlaceX*, *PushUntilClearY*, *RotateIntoPlaceY*, *SinglePushY*, *PushUntilClearX*, **PushTo**

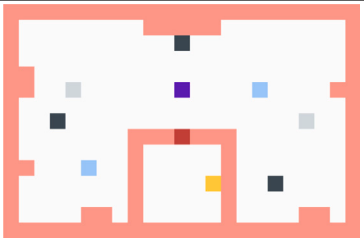
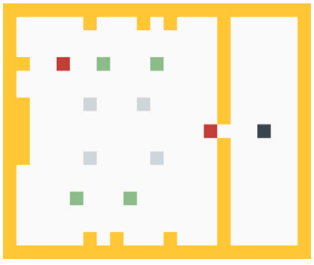
Table 8
Full list of non-hierarchical movement options.

Option name	Option description
RotateIntoPlace X/Y	Move clockwise or counterclockwise around a mobile object to get into place for pushing that object along the x/y-axis towards the target object.
Move/PushUntilAlignedX/Y	Move along the x/y-axis until mobile object has the same x/y-coordinate as the target object.
Move/PushUntilBlockedX/Y	Move along the x/y-axis towards the target object until the mobile object is blocked by another object.
Move/PushUntilClearX/Y	Move along the x/y-axis until there are no objects next to the mobile object.
SingleStep/PushX/Y	Move a single step along the x/y-axis towards the target object.
MoveUntilTouchX/Y	Move a along the x/y-axis until the mobile object is adjacent to the target object.

Table 9
Parameter values for value estimation.

Parameter	High-level description	Value
γ	Discount rate	0.90
α	Learning rate	0.05
c_{action}	Cost of taking each object-oriented action	0.01
D_s	Maximum action depth during MCTS simulation	0.0001
N_s	Number of simulations for MCTS	50
M_d	Maximum actions explored during action detection	30

Table 10
Descriptions for games. Game screen shows starting position of all object kinds. Object colors were randomized across participants. The colors written into the game rules in this table match the object colors shown in the game screen of this table.

Game name	Starting screen	Game rules
Collect the key		SpriteSet avatar > MovingAvatar color=Purple type2 > Passive color=LightBlue type3 > Passive color=COLOR3 type4 > Passive color=LightGray type5 > Passive color=Pink type6 > Passive color=Gold type7 > Passive color=Black InteractionSet type2 avatar > bounceForward type3 avatar > bounceForward type3 type2 > collectResource value=1 limit=1 type2 type7 > killWithScoreChange value=-1 avatar type7 > killSprite type3 type7 > killWithScoreChange value=-1 type4 type2 > killIfOtherHasMore res=type3 lim=1 type2 type4 > stepBack type6 avatar > killSprite TerminationSet SpriteCounter stype=type6 limit=0 win=True SpriteCounter stype=avatar limit=0 win=False
Destroy the blocks		SpriteSet avatar > MovingAvatar color=Black type2 > Passive color=LightGray type3 > Passive color=Purple type4 > Passive color=Pink type5 > Passive color=Green type6 > Passive color=Gold InteractionSet type2 avatar > bounceForward type3 avatar > bounceForward type3 type2 > killSprite type4 type2 > killSprite type5 type3 > killWithScoreChange value=1 TerminationSet SpriteCounter stype=type4 limit=0 win=True SpriteCounter stype=type3 limit=0 win=False
Fill the gaps		SpriteSet avatar > MovingAvatar color=Black type2 > Passive color=Red type3 > Passive color=LightGray type4 > Passive color=Green type5 > Passive color=Gold type6 > Passive color=Purple InteractionSet type2 avatar > bounceForward type3 avatar > nothing type3 type2 > transformTo stype=type6 type6 avatar > bounceForward avatar type4 > killSprite type2 type4 > killWithScoreChange value=-1 type4 type6 > killBoth TerminationSet SpriteCounter stype=type4 limit=0 win=True SpriteCounter stype=avatar limit=0 win=False

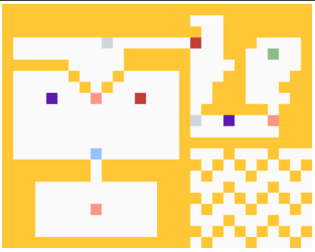
(*continued on next page*)

Appendix A. Theory inference details

Here we present the relevant details for implementing the likelihood function and structured prior that we described in the main paper.

As introduced in Section 2.2 of the main paper, we define a likelihood function using a graded transition probability, $P^*(s'|a, s, \theta)$. This assumes that all state transitions are equally important when calculating the likelihood. However, in practice, some states

Table 10 (*continued*).

Game name	Starting screen	Game rules
A transmutes, B destroys		SpriteSet avatar > MovingAvatar color=Green type2 > Passive color=Black type3 > Passive color=Purple type4 > Passive color=Red type5 > Passive color=LightGray type6 > Passive color=Pink type7 > Passive color=Gold InteractionSet type6 avatar > killWithScoreChange value=1 type3 avatar > bounceForward type4 avatar > bounceForward type2 type3 > transformTo stype=type6 type4 type3 > transformTo stype=type6 type2 type4 > killBoth type5 type4 > killBoth TerminationSet SpriteCounter stype=type6 limit=0 win=True SpriteCounter stype=type4 limit=0 win=False

contain more information than others. For example, some state transitions involve interactions that trigger multiple rules from the theory's interaction set. Intuitively, these states should weigh more heavily than states that don't involve any interaction information. Furthermore, state transitions that provide information about the termination conditions are relatively rare, and thus should also weigh more heavily. To account for this, we split the representation of s' into three components. s'_S contained all the information about object positions and features, while s'_I and s'_T contained information about which interactions and termination conditions were triggered as the player arrived in s'. We could then split the transition probability used for inference into three components, as follows:

$$P^*(s'|a,s,\theta) = P(s'_S|a,s,\theta)P(s'_I|a,s,\theta)P(s'_T|a,s,\theta) \tag{14}$$

$$P(s'_S|a,s,\theta) \propto \exp(w_S \mathbb{1}[s'_S = \theta_S(a,s)]) \tag{15}$$

$$P(s'_I|a,s,\theta) \propto \exp(w_I \mathbb{1}[s'_I = \theta_I(a,s)]) \tag{16}$$

$$P(s'_T|a,s,\theta) \propto \exp(w_T \mathbb{1}[s'_T = \theta_T(a,s)]), \tag{17}$$

where $\theta_S(a,s)$, $\theta_I(a,s)$, and $\theta_T(a,s)$ represent the object positions, triggered interactions, and triggered termination conditions predicted by θ. The scalar parameters w_S, w_I, and w_G are weights determining the relative influence of information about the object positions, interactions, and termination conditions. These were fit by hand to the values in Table 6 to reflect the approximate rarity of each kind of event in the game data.

We also defined a structured prior over theories by converting each theory to a set of predicates and defining a series logical expressions over those predicates. In essence, we have defined a typed, second-order logical system for describing VGDL theories. Table 3 describes the predicates that we use in more detail. Each predicate in Table 3 takes in typed variables as inputs and returns a variable of a specific type. Table 4 shows the potential values for each variable type. Every theory in the subset of VGDL that we consider here represents a unique grounding of these typed predicates to specific values. Similarly, every unique grounding of predicates represents one VGDL theory (though not necessarily a unique VGDL theory). Thus, when taken together, Tables 3 and 4 define the full space of possible theories considered by the model.

The inductive biases we consider in the main paper are all logical expressions composed of the predicates in Table 3. For example, all syntactically valid VGDL theories must have at least one player-controllable sprite type in the sprite set. We can represent this constraint with the logical expression below:

$$\exists s \in \textbf{SpriteType} \text{ s.t. } HasSpriteType(s) \wedge SpriteFeature(s) == \text{MovingAvatar}$$

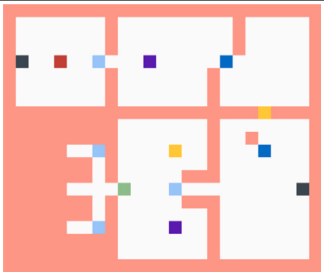
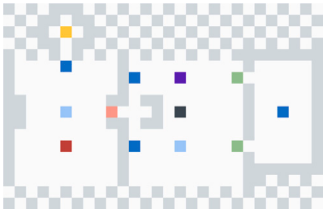
Similarly, we can represent the constraint that the agents involved in each interaction must exist with the following pair of expressions:

$$\forall i \in \textbf{Interaction} \cdot HasInteraction(i) \Rightarrow$$

$$HasSpriteType(InteractionAgent(i))$$

Table 5 describes the set of syntactic biases used in this work. All of the syntactic biases and the majority of the semantic biases consist of relatively simple logical expressions. However, as described in Section 2.3 of the main paper, the attainability and purpose biases require generating causal chains. While it is possible to encode the process for generating these chains as a logical expression using the predicates in Table 3, the resulting expression would be too long to reasonably write out here. For brevity, we

Table 11
Continuation of Table 10.

Game name	Starting screen	Game rules
A pushes, B freezes		SpriteSet avatar > MovingAvatar color=Red type2 > Passive color=LightBlue type3 > Passive color=Purple type4 > Passive color=Gold type5 > Passive color=Blue type6 > Passive color=Green type7 > Passive color=Pink type8 > Passive color=Black InteractionSet type2 avatar > killWithScoreChange value=1 type3 avatar > bounceForward type4 avatar > bounceForward type5 type3 > bounceForward type6 type3 > bounceForward type5 type4 > transformTo stype=type7 type6 type4 > transformTo stype=type7 avatar type8 > killSprite TerminationSet SpriteCounter stype=type2 limit=0 win=True SpriteCounter stype=avatar limit=0 win=False
A opens everything		SpriteSet avatar > MovingAvatar color=Gold type2 > Passive color=LightBlue type3 > Passive color=Pink type4 > Passive color=Red type5 > Passive color=Purple type6 > Passive color=Black type7 > Passive color=Green type8 > Passive color=Blue type9 > Passive color=LightGray InteractionSet type8 avatar > killWithScoreChange value=1 type2 avatar > bounceForward type5 avatar > bounceForward type4 type2 > collectResource value=1 limit=1 type3 type2 > killIfOtherHasMore res=type4 lim=1 type2 type3 > stepBack type6 type2 > collectResource value=1 limit=1 type7 type2 > killIfOtherHasMore res=type6 lim=1 type2 type7 > stepBack type5 type7 > killSprite TerminationSet SpriteCounter stype=type8 limit=0 win=True SpriteCounter stype=type5 limit=0 win=False

(*continued on next page*)

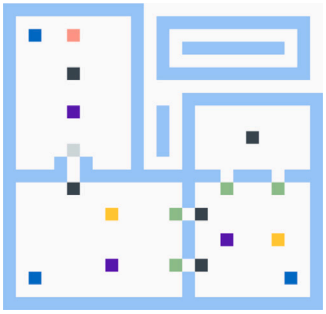
instead generate these causal chains with the algorithms shown in Algorithm 1 and Algorithm 2 and define two helper functions $forwardChain(k)$ and $reverseChain(g)$ that return the set of rules and goals in each forward chain or rules and objects in each reverse chain, respectively.

Finally, we approximate the posterior described in Section 2 of the main paper using a variational inference approach known as *Discrete Particle Variational Inference* (DPVI). Variational approaches approximate a posterior by finding the distribution Q in a family of distributions $\mathbf{Q}$ that maximizes the negative variational free energy, $L[Q]$. In discrete particle VI, each distribution in $\mathbf{Q}$ is represented by a set of weighted particles. Saeedi et al. (2017) describe an optimization algorithm for finding Q that performs coordinate ascent from a random set of starting particles. They prove that the negative free energy is maximized by finding the K particles with the highest likelihood scores and associating those particles with weights p_k as follows:

$$p_k = Z_Q^{-1} \frac{f(\theta_k)}{V_k}, \tag{18}$$

where $f(\theta_k)$ is the likelihood of the theory θ_k associated with the kth particle, V_k is the number of times that the kth particle appears in Q, and $Z_Q = \sum_k \frac{f(\theta_k)}{V_k}$ is a normalizing constant.

Table 11 (*continued*).

Game name	Starting screen	Game rules
A creates coins		SpriteSet avatar > MovingAvatar color=Pink type2 > Passive color=Purple type3 > Passive color=LightGray type4 > Passive color=Gold type5 > Passive color=Green type6 > Passive color=Black type7 > Passive color=LightBlue type8 > Passive color=Blue InteractionSet type6 avatar > killWithScoreChange value=1 type2 avatar > bounceForward type4 avatar > bounceForward type3 type2 > transformTo stype=type6 type5 type2 > transformTo stype=type6 type2 type6 > killSprite type5 type4 > transformTo stype=type7 type6 type4 > transformTo stype=type7 avatar type8 > killSprite TerminationSet SpriteCounter stype=type6 limit=0 win=True SpriteCounter stype=avatar limit=0 win=False

Algorithm 1 GenerateForwardChains(θ)

```
Initialize ForwardChains, VisitedElements, q
Add avatar of θ to q
start ← avatar
while |q| > 0 do
  e ← Pop(q)
  if e ∉ VisitedElements then
    Add e to VisitedElements
    Add e to ForwardChains(start)
    if Type(e) == sprite then
      Prepend interactions where e is agent, patient, or resource to q
    else if Type(e) == interaction then
      Prepend agent, patient, resource, or stype sprites to q
      if InteractionEffect(e) pushes something then
        Prepend all interactions involving pushed sprite
      else if InteractionEffect(e) destroys something then
        Prepend all goals involving destroyed sprites
      else if InteractionEffect(e) requires resource then
        Prepend all interactions involving collecting resource
      end if
    end if
  end if
end while
return ForwardChains
```

The original version of this algorithm often only explores a limited subset of the distributional space, thus for the current work we used a modified version of the algorithm that identifies multiple plausible Qs from different starting points, and then takes the highest weighted particle from each. This empirically produces more diverse posterior approximations. Additionally, the original algorithm requires evaluating all single step changes to each theory, which drastically slows the algorithm for larger theory spaces. Instead our algorithm only evaluates a random sample of N_S single step changes. When fixing the presence of particular rules in generating the model simulations for the prediction task we also ensured that all the random starting theories contained the given rule r and that all single step changes we sampled maintained the inclusion of this rule. We present our modified algorithm in Algorithm 3, along with a list of parameters and values in Table 6.

In prior work, we found that combining Monte Carlo Tree Search with Upper Confidence bound for Trees (MCTS-UCT) over object-oriented actions with value estimation techniques from the RL literature was sufficient for modeling human planning in

Table 12
Full list of questions used to assay semantic biases. Bold text in rules indicates difference between bias upheld and bias violated conditions.

Bias name	Question pair	Game	trigger	Bias upheld rule	Bias violated rule
Non-arbitrary limits	1	Fill the gaps	avatar/type2	How likely is it that "Win when there are **ZERO** of type3" is a rule?	How likely is it that "Win when there are **TWO** of type3" is a rule?
Non-arbitrary limits	2	A opens everything	type2/type4	How likely is it that "Win when there are **ZERO** of type8" is a rule?	How likely is it that "Win when there are **TWO** of type8" is a rule?
Non-arbitrary limits	3	A pushes, B freezes	type3/type6 or type4/type6	How likely is it that "Win when there are **ZERO** of type2" is a rule?	How likely is it that "Win when there are **ONE** of type2" is a rule?
Attainability	1	Fill the gaps	avatar/type2	Imagine the rule "type3 turns type4 into type6." How likely is it that "avatar **PUSHES** type3" is also a rule?	Imagine the rule "type3 turns type4 into type6." How likely is it that "avatar **DESTROYS** type3" is also a rule?
Attainability	2	Destroy the blocks	avatar/type2	Imagine the rule "type3 destroys type4 for 1 point." How likely is it that "avatar pushes **TYPE3**" is also a rule?	Imagine the rule "type3 destroys type4 for 1 point." How likely is it that "avatar pushes **TYPE5**" is also a rule?
Attainability	3	A creates coins	avatar/type6	Imagine the rule "type4 and type5 kill each other." How likely is it that "avatar **PUSHES**" type4 is also a rule?	Imagine the rule "type4 and type5 kill each other." How likely is it that "avatar **DESTROYS** type4" is also a rule?
Consistency	1	Collect the key	type2/type7 or type3/type7	How likely is it that "type2 **REMOVES A POINT BY DESTROYING**" type7 is a rule?	How likely is it that "type2 **TRANSFORMS** type7 **INTO TYPE5**" is a rule?
Consistency	2	A transmutes, B destroys	type5/type4	How likely is it that "type2 **TRANSFORMS TYPE3 INTO TYPE6**" is a rule?	How likely is it that "type2 **AND TYPE4 DESTROY EACH OTHER**" is a rule?
Consistency	3	A pushes, B freezes	avatar/type4	How likely is it that "type3 **PUSHES** type6" is a rule?	How likely is it that "type3 **TRANSFORMS** type6 **INTO TYPE7**" is a rule?
Informative-ness	1	A opens everything	avatar/type8	How likely is it that "**WIN** when there are zero type8 left" is a rule?	How likely is it that "**LOSE** when there are zero type8 left" is a rule?
Informative-ness	2	A transmutes, B destroys	type2/type3	How likely is it that "**WIN** when there are zero type6 left" is a rule?	How likely is it that "**LOSE** when there are zero type6 left" is a rule?

(*continued on next page*)

simple grid-based video games (Pouncy et al., 2021). For the games in this previous work, a simple Breadth-First Search (BFS) over directional actions was enough to identify the small set of objects that could be interacted with in a reasonable number of steps. The resulting set of object-oriented actions was also small enough that we could use the standard version of MCTS-UCT. However, the games in the current work involved objects that could be pushed into one another. This created two problems: larger levels, and a greater number of object-oriented actions.

Table 12 (*continued*).

Bias name	Question pair	Game	trigger	Bias upheld rule	Bias violated rule
Informative-ness	3	A pushes, B freezes	avatar/type2	How likely is it that "**WIN** when there are zero type2 left" is a rule?	How likely is it that "**LOSE** when there are zero type2 left" is a rule?
Purpose	1	Destroy the blocks	type2/type3	How likely is it that there is a rule you haven't seen yet involving **TYPE5**?	How likely is it that there is a rule you haven't seen yet involving **TYPE2**?
Purpose	2	A transmutes, B destroys	type4/type5	How likely is it that there is a rule you haven't seen yet involving **TYPE2**?	How likely is it that there is a rule you haven't seen yet involving **TYPE5**?
Purpose	3	A creates coins	type2/type3	How likely is it that there is a rule you haven't seen yet involving **TYPE5**?	How likely is it that there is a rule you haven't seen yet involving **TYPE6**?
Win/Lose	1	Collect the key	avatar/type6 or avatar/type7	How likely is it that "**LOSE** when there are zero avatar left" is a rule?	How likely is it that "**WIN** when there are zero avatar left" is a rule?
Win/Lose	2	Fill the gaps	avatar/type4	How likely is it that "**WIN** when there are zero type3 left" is a rule?	How likely is it that "**LOSE** when there are zero type3 left" is a rule?
Win/Lose	3	Destroy the blocks	type2/type3	How likely is it that "**WIN** when there are zero type5 left" is a rule?	How likely is it that "**LOSE** when there are zero type5 left" is a rule?

Table 13
P-values for t-tests comparing lesioned agents to the syntactic agents in Fig. 8. Red text indicates cells where there was significant evidence of difference ($p < 0.05$). P-values were corrected within columns using a Bonferroni correction.

Bias	Arb. Goals vs. Syntactic	No Attain. vs. Syntactic	No Cons. vs. Syntactic	No Inf. Pts vs. Syntactic	No Purpose vs. Syntactic	No W/L vs. Syntactic
Non-arbitrary goals	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**
Attainability	**p<0.01**	p=0.20	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**
Consistency	**p<0.01**	**p<0.01**	**p=0.01**	**p<0.01**	**p<0.01**	**p<0.01**
Informativeness	**p<0.01**	**p<0.01**	**p<0.01**	**p=0.03**	**p<0.01**	**p<0.01**
Purpose	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**
Win/Lose	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	**p<0.01**	p=1.00

Table 14
P-values for t-tests comparing lesioned agents to the semantic agents in Fig. 8. Red text indicates cells where there was significant evidence of difference ($p < 0.05$). P-values were corrected for multiple comparison within column using a Bonferroni correction.

Bias	Arb. Goals vs. Semantic	No Attain. vs. Semantic	No Cons. vs. Semantic	No Inf. Pts vs. Semantic	No Purpose vs. Semantic	No W/L vs. Semantic
Non-arbitrary goals	**p<0.01**	p=1.00	p=1.00	p=1.00	p=1.00	p=0.55
Attainability	p=1.00	**p<0.01**	p=1.00	p=1.00	p=1.00	p=1.00
Consistency	p=1.00	p=1.00	**p<0.01**	p=1.00	p=1.00	p=1.00
Informativeness	**p<0.01**	**p<0.01**	p=1.00	**p<0.01**	p=0.26	p=0.36
Purpose	p=1.00	p=1.00	p=1.00	p=1.00	**p<0.01**	p=1.00
Win/Lose	p=0.46	**p<0.01**	p=1.00	p=0.54	p=1.00	**p<0.01**

Table 15
Comparison of CFA models. Parentheses represent 90% confidence interval for RMSEA.

Model name	Model description	χ^2	Deg. of freedom	RMSEA	Comparative fit index
Independent bias factors	Zero correlation between bias weights.	698.6	136	0.130 (0.121, 0.140)	0.57
All bias factor correlations	Correlations for all biases.	374.7	122	0.092 (0.082, 0.103)	0.81
Sig. bias factor correlations	Only significant correlations between biases.	378.6	125	0.091 (0.081, 0.102)	0.80

Table 16
Parameters of mixed effects models for competence, directedness and humanness (for both "more human = more competent" and "more human = less competent" participants). Each cell represents parameter estimate and associated p-value. "N/A" indicates that parameter was not included in best-fitting model. Red and green text indicate negative and positive parameters that were significant at the $P < 0.05$ level, respectively.

Parameter	Competence Ratings	directedness Rating	humanness Rating (More Hum. = More Comp.)	humanness Rating (More Hum. = Less Comp.)
Intercept	**58.73** (**p < 0.01**)	**64.44** (**p < 0.01**)	**55.95** (**p < 0.01**)	**60.67** (**p < 0.01**)
Pct. stuck runs	**−21.43** (**p < 0.01**)	**−19.53** (**p < 0.01**)	−12.94 (p = 0.08)	**13.82** (**p = 0.04**)
Pct. success runs	**12.70** (**p < 0.01**)	**12.17** (**p < 0.01**)	**10.96** (**p = 0.02**)	**−12.83** (**p < 0.01**)
Agent type (Human v. Semantic)	**10.58** (**p < 0.01**)	**8.51** (**p < 0.01**)	**10.60** (**p < 0.01**)	**−9.01** (**p < 0.01**)
Agent type (Semantic v. Syntactic)	0.72 (p = 0.65)	1.03 (p = 0.51)	6.10 (p = 0.07)	0.66 (p = 0.84)
Num. Avatar/Death	**−5.64** (**p < 0.01**)	**−4.75** (**p < 0.01**)	**−5.84** (**p < 0.01**)	**5.77** (**p < 0.01**)
Num. Avatar/Rew.	0.00 (p = 0.99)	0.64 (p = 0.12)	−0.12 (p = 0.89)	−0.08 (p = 0.92)
Num. Mobile/Wall	**−0.42** (**p < 0.01**)	**−0.42** (**p < 0.01**)	**−0.37** (**p = 0.02**)	**0.42** (**p < 0.01**)
Num. Tool/Target	0.77 (p = 0.19)	0.29 (p = 0.61)	0.79 (p = 0.52)	0.31 (p = 0.78)
Num. Tool/Tool	0.19 (p = 0.09)	**0.35** (**p < 0.01**)	0.06 (p = 0.82)	−0.08 (p = 0.69)

In order to give the players enough space to push tools around each other, objects in the current games had to be further apart. As the search time of the BFS algorithm increases exponentially with the average space between objects, these larger levels made the original object-detector intractable for the games in the current work. To get around this, we replaced this BFS algorithm with a hierarchical planner. We provided this planner with a number of low-level options for moving and pushing objects in a 2-dimensional grid space. Crucially, these low-level options were agnostic to the distance between objects (e.g., "move left until you touch a target object", "push a block up until it is in the same row as a target object", etc.). Some of these options were hierarchically defined in terms of other options. For example, pushing an object to the left required first moving the avatar so that it was to the right of the pushed object. Tables 7 and 8 provide a complete list of these non-hierarchical and hierarchical options, respectively.

To identify potential object-oriented actions, we created a function $GetNearestObjects$ that generated a list of every kind of object in a game that could be moved and identified the 2 instances of each object kind that were nearest to those mobile objects. Finally, for each of these potential object interactions, the detector would start with either the $MoveTo$ (if the interaction involved the avatar) or $PushTo$ (if the interaction involved a non-avatar mobile object) option defined in Table 7 and try to find a combination of options that would produce the target interaction by using the $GetNextAlternative()$ function to randomly iterate through the alternatives for that option. Algorithm 4 provides pseudo-code for this process.

The ability to push objects into one another also increased the average number of object-oriented actions available from each state. This exponentially increased the size of the state space. Classically, MCTS-UCT simulates actions using a uniform random policy to explore states until all actions from a state have been explored at least once. The algorithm then switches to selecting the action with the highest upper confidence bound. This random initial policy means that MCTS-UCT fails to produce reasonable value

Algorithm 2 GenerateRevChains(θ)

```
Initialize RevChains, VisitedElements, q
for goal in θ do
  Add goal to q
  start ← goal
  while |q| > 0 do
    e ← Pop(q)
    if e ∉ VisitedElements then
      Add e to VisitedElements
      Add e to RevChains(start)
      if Type(e) == goal then
        Prepend interactions that trigger e
      else if Type(e) == interaction then
        Prepend agent, patient, resource, or stype sprites to q
        if InteractionEffect(e) pushes something then
          Prepend all interactions that block or destroy the pushing object
        end if
      else if Type(e) == sprite then
        if e is mobile then
          Prepend all interactions with all obstacles that e can destroy or move
        else if e is a resource then
          Prepend all interactions involving collecting e
        end if
      end if
    end if
  end while
end for
return RevChains
```

estimates until each state has been simulated many times. In theory the estimates from this algorithm can be improved by increasing the number of simulations used during value estimation, however the number of simulations required proved to be computationally intractable for our task. Instead, we used the leaf parallelization strategy described by Chaslot, Winands, and van den Herik (2008). Leaf parallelization speeds up MCTS by simulating multiple actions in parallel whenever a new state is reached. This allows the algorithm to get to the second, upper-confidence bound selection policy more quickly.

In addition, we further improved the early MCTS value estimates by using the causality graphs that we generated to evaluate the purpose and attainability biases to approximate how relevant interactions were for the different goals in a game theory. Rather than selecting actions at random, our updated MCTS algorithm preferentially explored interactions that were closer to achieving rewarding goals. This helped ensure that early value estimates for recently expanded nodes more accurately represented the maximum potential values for that node. Aside from these changes, we used the same value estimation process and planning parameters that we used in the prior work. The new parameters introduced by the hierarchical action detector were fit to maximize the performance of agents that had been given access to ground truth theories for each game. Table 9 provides a list of the parameters used here. See Pouncy et al. (2021) for additional details about the MCTS-UCT algorithm and the value estimation process.

Appendix B. Additional task details

In this section we present additional details about the question pairs that we used to assay the biases described in the main paper. As mentioned in the main paper, we had participants play seven games in random order. Each question pair was displayed within a few steps of a participant triggering a particular interaction. Table 12 provides a full list of the question pairs that participants could trigger. We also include the bias related to each question pair, the game in which the pair occurred, the triggering condition for each pair, and the bias upheld and bias violated question within each pair. For context, Tables 10 and 11 provide the starting positions and rule sets for each game. The types in each question pair refer to the sprite types in the rules for the corresponding game. In order to avoid confusing participants, all instances of "typeX" in the questions were replaced with colored squares that matched the appropriate object kind in each game before presenting questions to participants.

Each question pair was designed to assay a particular bias. As such, question pairs that assay the same bias have similar structure. Four of the biases involved asking about the likelihood of an unobserved rule or goal based on the rules and goals that the participant had observed so far. The non-arbitrary goals question pairs compared the likelihood of goals involving removing all of some object kind rather than an arbitrary number of some object kind. Each informativeness question pair was triggered by discovering a rewarding object kind, and involved comparing the likelihood that collecting all the rewarding objects would win the game rather than lose it. Similarly, each win/lose question pair was triggered by discovering either a winning or losing goal of the game and

Algorithm 3 DiscreteParticleVariationalInference(N_A, K, ϵ)

```
/* INPUT: Number of variational distributions to approximate (N_A), */
/* Number of particles for each variational distribution (K), */
/* Variational free energy threshold (ε) */
for a = 1 : N_A do
  Initialize the ath variational distribution Q_a with K particles
  L[Q] ← ∞
  for θ_k ∈ Q_a do
    p_{a,k} ← Z_Q^{-1} f(θ_k)/V_k
  end for
  L[Q'] ← log(Z_Q)
  while |L[Q] − L[Q']| > ε do
    L[Q] ← L[Q']
    Initialize new particles P'_a and weights p'_a
    for k = 1 : K do
      for d ∈ SubSample(Predicates) do
        for v ∈ Domain(Output(p)) do
          Copy particle Q_{a,k}
          Set component d of new particle to v
          Add new particle to P'_a
          Set weight for new particle to Z_Q^{-1} f(θ_k)/V_k
        end for
      end for
    end for
    Q'_a ← N_k unique particles from P'_a with highest weights
    for θ_k ∈ Q'_a do
      p_{a,k} ← Z_Q^{-1} f(θ_k)/V_k
    end for
    L[Q'] ← log(Z_Q)
  end while
end for
Q ← Top weighted particle from each Q'_a
for θ_k ∈ Q do
  p_k ← Z_Q^{-1} f(θ_k)/V_k
end for
return Q
```

then asking whether another potential goal would be more likely to be a winning condition or a losing condition. Finally, each consistency question pair was triggered when the participant learned how to use a given tool, and involved inferring how that tool would interact with a novel object.

The attainability question pairs all involved imagining learning a new rule involving two unexplored object kinds and then evaluating a second rule based on the imagined rule. In the bias upheld question the second rule would make the imagined rule attainable, while in the bias violated question the second rule would not make the imagined rule attainable. For example, the imagined rule "the purple block destroys the pink block", is definitely attainable if the avatar can push the purple block, but may or may not be attainable if the avatar can push an unrelated green block.

Finally, the purpose question pairs all involved asking about the likelihood of additional unobserved rules about a given object kind. Each of these question pairs was triggered when the participant had discovered a purpose for one kind of object (A) but had yet to discover a purpose for another kind of object (B). In the bias violated question, participants were asked how likely it was that there were additional rules involving A, while in the bias upheld question they were asked about additional rules involving B.

Appendix C. Additional results

Fig. 8 shows the average response differences for the lesioned semantic agents. As in Fig. 4, the y-axis represents the difference between bias upholding and bias violating questions, averaged across all three question pairs for each semantic bias. Each cell in Fig. 8 represents the question pairs associated with a different semantic bias, while the x-axis represents the type of agent evaluated. For comparison, the syntactic agents are on the far left of each x-axis, while the regular semantic agents are on the far right. Table 13 shows p-values for independent t-tests comparing each type of lesioned agent to the syntactic agents, while Table 14 shows the p-values for t-tests comparing lesioned agents to the regular semantic agents. For the attainability, informativeness, and win/lose

Algorithm 4 DetectObjectOrientedActions(θ, s)

```
Initialize ObjectOrientedActions
Add avatar of θ to q
q ← GetNearestObjects(θ, s)
while |q| > 0 do
  t ← Pop(q)
  h ← ResetHierarchicalOptions()
  Initialize current action path p
  CurrentAlternatives ← GetNextAlternative(h)
  i = 0
  while |CurrentAlternatives| > 0 and i < M_d do
    i = i + 1
    CurrentLevel, o ← Pop(CurrentAlternatives)
    TriggeredInteractions, s, a ← TakeOption(o)
    p ← p + a
    if |TriggeredInteractions| > 0 then
      if t in TriggeredInteractions then
        Add p to ObjectOrientedActions
        Clear CurrentAlternatives
      else
        Undo all changes to p and S at current level
      end if
    else if i == M_d then
      Undo all changes to p and S at current level
    end if
  end while
end while
return ObjectOrientedActions
```

biases, only the lesioned agents missing the associated bias showed no evidence of difference from the syntactic agents. This suggests that these question pairs uniquely assayed their associated biases.

In contrast, the arbitrary goals, consistency, and purpose lesioned agents still produced response differences that are substantially higher than the syntactic agent. The fact that the syntactic agent produced responses that are substantially lower than these lesioned agents suggests that the question pairs associated with these three biases may also partially assay the presence of other biases. However, Table 14 shows all three of these lesioned agents still produced substantially lower responses than the regular semantic agents, suggesting that these question pairs were at least predominantly assaying their intended biases.

Appendix D. Confirmatory factor analysis details

As mentioned in Section 3.3 of the main paper, we used a Confirmatory Factor Analysis (CFA) to efficiently estimate participant's bias weights. We tested three different CFA models to find the best fit to the data. All three models included six latent factors (one for each semantic bias). Each factor was estimated using the three question pairs associated with each bias. We then tested a model with completely independent latent factors, a model with correlations between all latent factors, and a model with only the significant correlations between latent factors. Table 15 shows the χ^2, Root Mean Squared Error of Approximation (RMSEA) and Comparative Fit Index (CFI) statistics for these three models. The large decrease in χ^2 between the zero correlations model and the all correlations model suggests that adding the factor correlations improved model fit. There was no meaningful difference between the model with all factor correlations and the model with only the significant correlations. We chose the model with only the significant correlations as it produced marginally lower RMSEA and CFI values.

Fig. 9 shows distributions of the fitted weights for each semantic bias found by the best-fitting CFA. Qualitatively we see evidence of some individual variation in each parameter. For the non-arbitrary goals, informativeness, and win/lose biases, the human fitted parameters skew towards higher values, while for the attainability, consistency, and purpose biases the fitted parameters skew towards low values. However, we see a range of values for all biases suggesting some potential variability in the population.

Fig. 10e shows the correlation between participants' scores on each of the question sets and the performance of their yoked agents. The x-axis here reflects participants' difference scores, while the y-axis reflects the difference scores of the agents with that participant's fitted weight parameters. Since the model parameters for the syntactic, lesioned, and semantic agents were identical across participants, the only difference between individual instances of these agent types was the order in which rules and questions were observed. We found no significant correlations between the syntactic agents, lesioned, or semantic agents and their yoked human participants. This suggests that individual variability cannot be accounted for with differences in question order under these

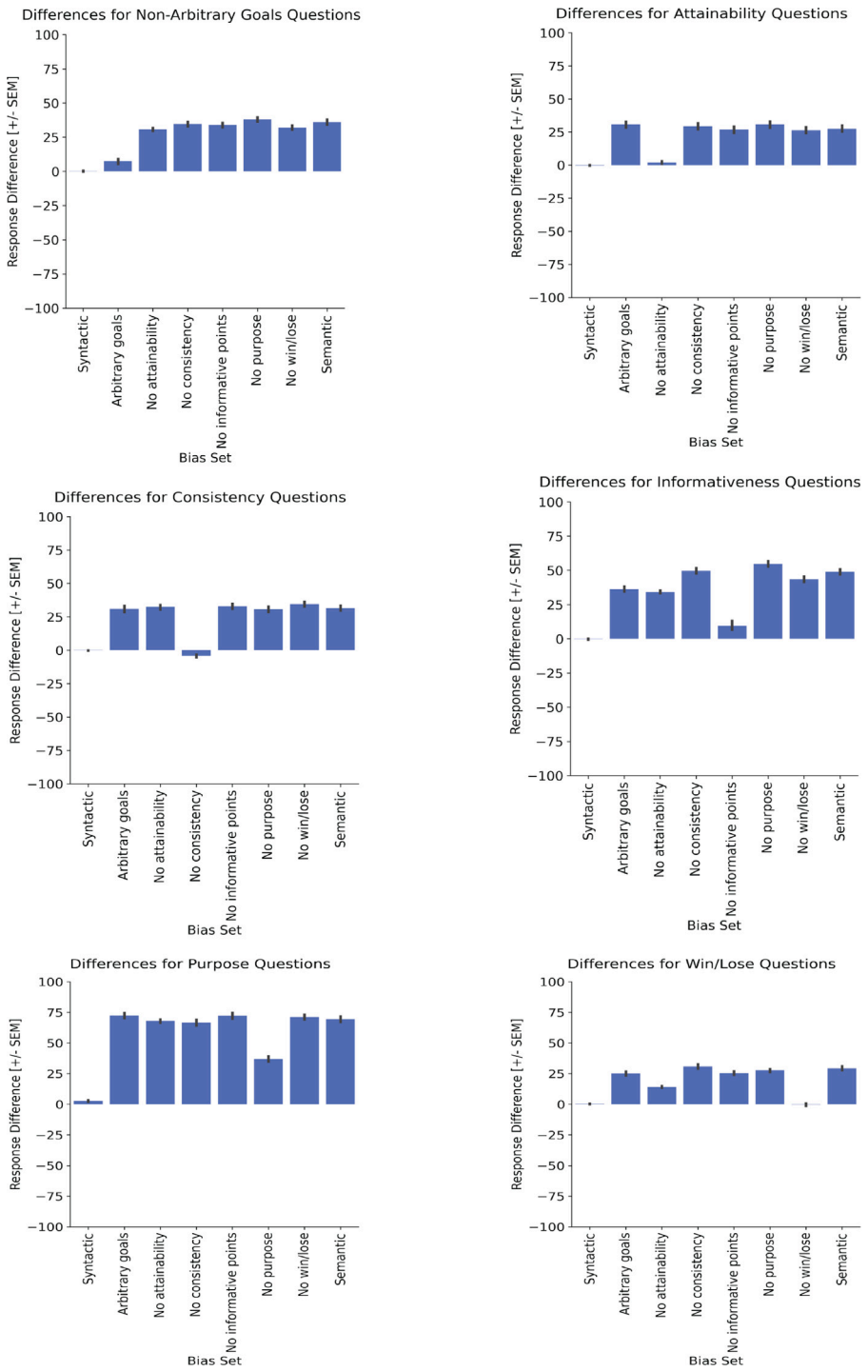


Fig. 8. Average response difference for lesioned agents. *Y*-axis shows the response difference between the bias upheld and bias violated questions for each question pair. *X*-axis represents agent type. For comparison, the far left agent in each figure is the syntactic agent, while far right is the semantic agent. Middle agents have all semantic biases but one. Error bars show standard error of the mean.

models. In contrast, there were significant correlations between the fitted bias agents and their human participants for the non-arbitrary goals, attainability, informativeness, and win/lose biases, suggesting that at least some of the variance in human responses could be accounted for with differences in latent bias weights.

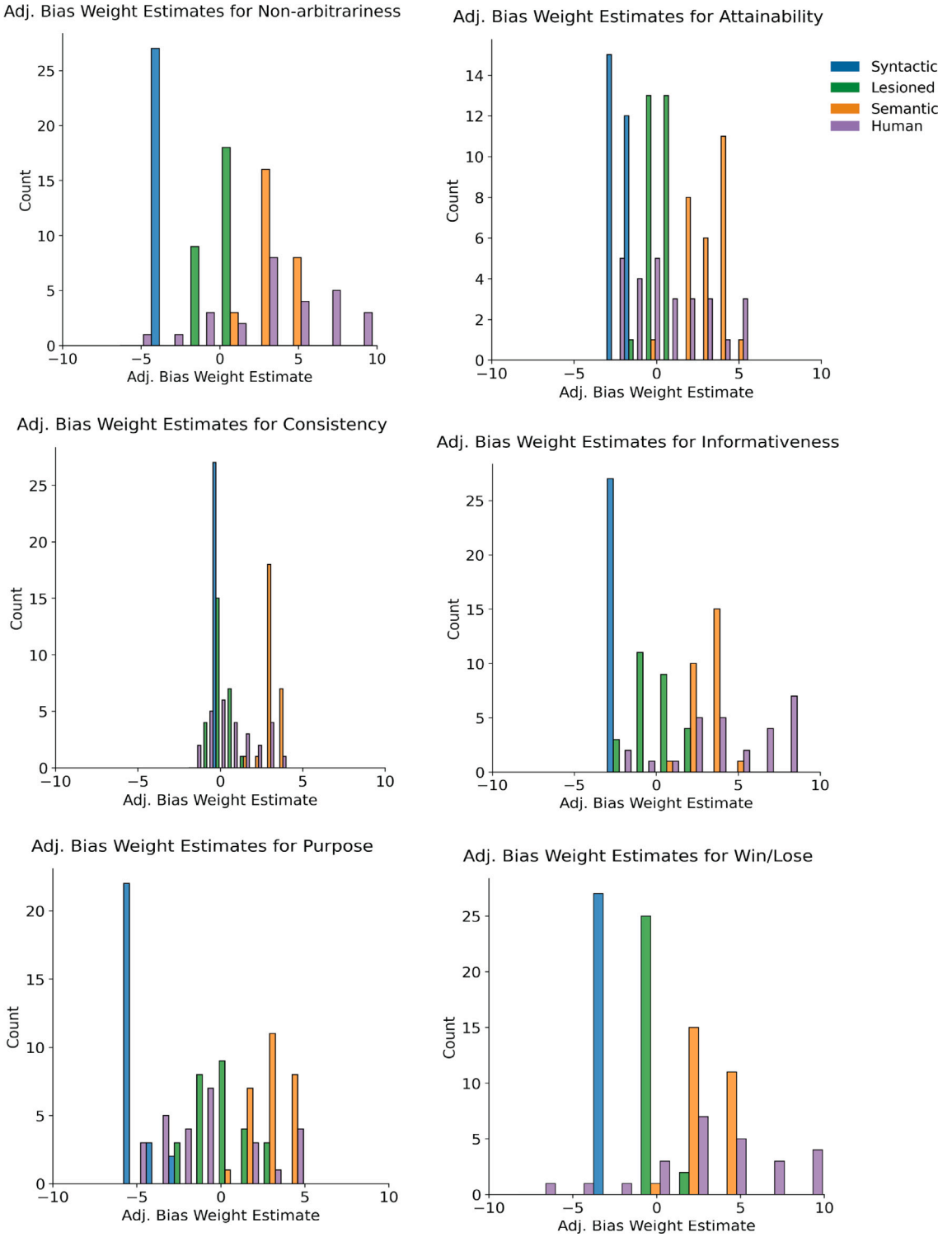


Fig. 9. Frequency counts of fitted bias weight parameters $\mathbf{w_b}$ for each of the six additional biases. Each cell corresponds to one bias. The horizontal axis represents the weight value, while the vertical axis represents the number of agents with that weight value. Color indicates agent type.

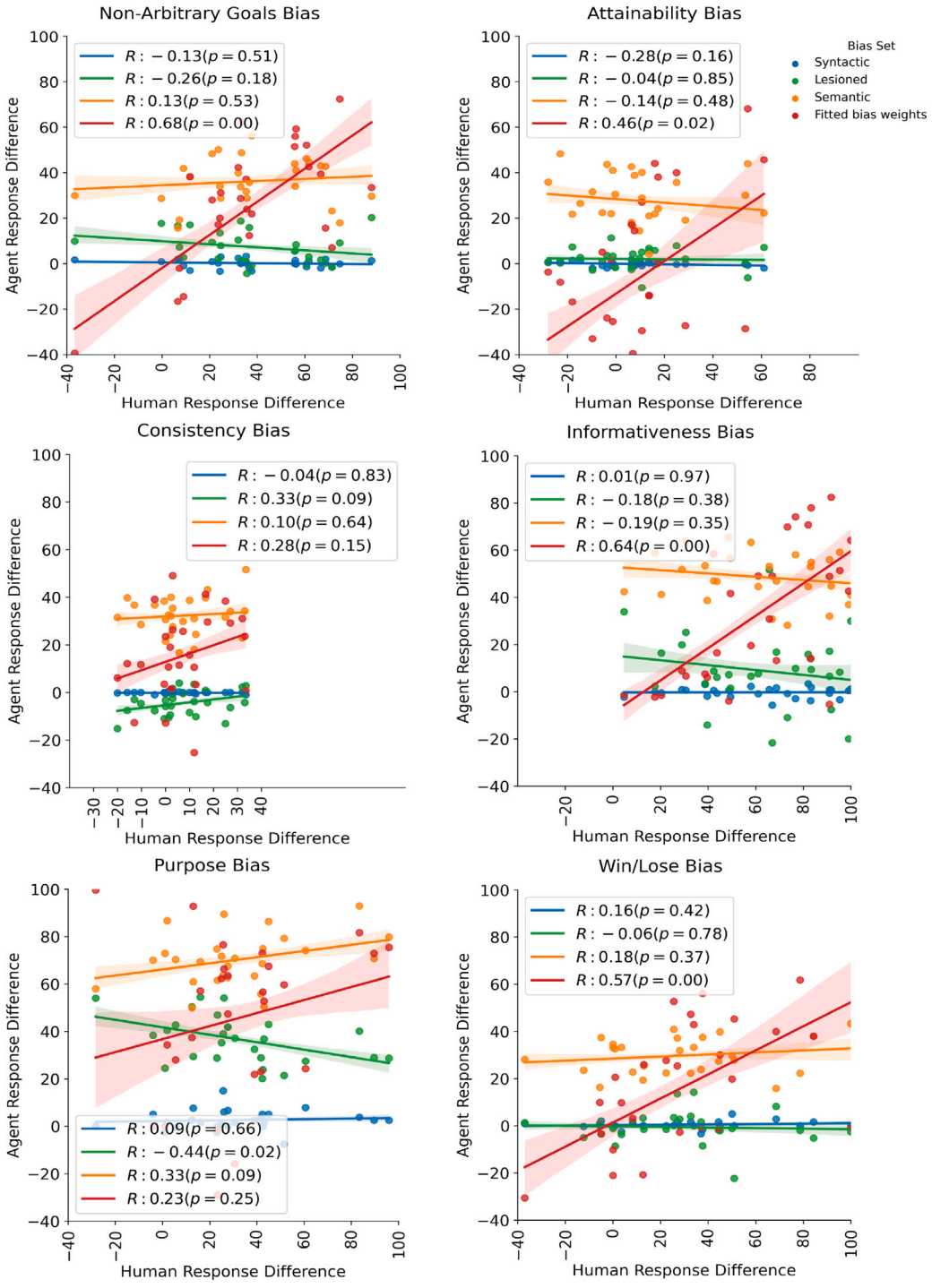


Fig. 10. Correlations between agent and human response differences for each semantic bias. X-axis represents the human response difference for each question pair associated with a given bias. Y-axis represents the response difference from the agent whose actions were yoked to a given participant. Error bars reflect 1 standard error of the mean. Color represents which bias weight set an agent had. The quantities next to each element in the legend reflect Pearson correlations, while the numbers in parentheses represent p-values.

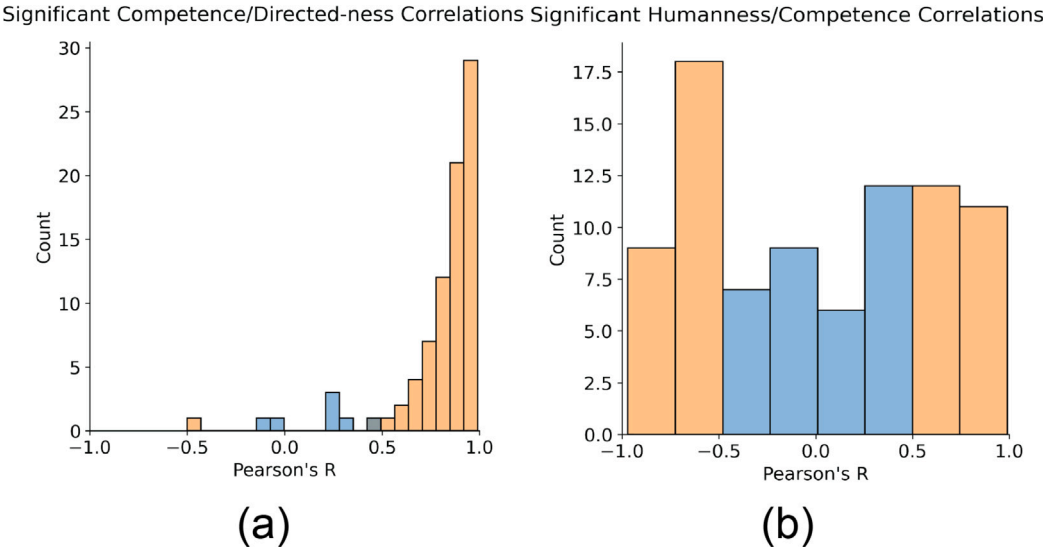


Fig. 11. Distribution of correlations between (a) competence and directedness, and (b) competence and humanness ratings within participants. Orange bars represent correlations that were significant at the $p < 0.05$ level.

Appendix E. Additional details about independent raters

We recruited 87 participants using the CloudResearch interface and vetting process mentioned in Section 3 of the main paper. Participants were told that they were going to play a series of games and that their goal was to earn as many points as possible on these games. The bonus payment structure and set of games used were the same as in the original planning experiment. However, for the rating experiment, participants were also told that after each game, they would be asked to watch videos of three other agents learning to play the same game. Participants were informed that some of these agents would be computers and some would be humans. After watching each video, participants were asked to rate the agents on a series of scales. The tutorial for this experiment included sample videos and ratings so that participants could familiarize themselves with the interface before play. We removed two participants who failed to successfully complete three or more runs.

The structure of the rating experiment consisted of a brief tutorial followed by seven blocks, one for each of the seven games from original planning task. Although we collected data for all seven games, one of the seven games that we evaluated (the "Destroy the blocks" game), produced no difference in success rate or proportion of interactions between the artificial agents and the human participants. This suggests that this game was not useful as a diagnostic tool for differentiating agent behavior. Thus we omitted this game from the results presented for this experiment. As before, the games were presented in random order.

In each block we first asked participants to play three runs of the game in order to familiarize themselves with the game's structure. Then we presented them with three videos of other agents learning to play the same game. There was always one video of an syntactic agent, semantic agent, and human participant, but the order of presentation varied from block to block. Videos were selected at random from the 27 instances of each agent type. To avoid confusion, the colors of the objects in each video were changed to match the colors of the objects that the rater saw during their play through. Since humans often paused between decisions while agents did not, we generated the videos using a fixed pace of 5 actions per second for both agents and humans.

After watching each video, participants were asked to answer the following three questions: "Was the agent in this video a human or a computer?", "How competent was the agent you just watched?", and "How purposeful were the agent's actions?". Participants answered each question using a sliding scale from 0 ("Definitely a computer", "Not competent", or "Not purposeful/random") to 100 ("Definitely a human", "Very competent", or "Very purposeful"). We will refer to these three ratings as "humanness", "competence", and "directedness". To ensure that participants had watched the video before rating, the button to continue the task would only appear after each video had been playing for at least 10 s.

As mentioned in the main paper, participants interpreted the humanness rating differently. Fig. 11 shows the distribution of Pearson correlations between each participant's humanness and competence ratings.

In addition to the analyses presented in the main paper, we also fit mixed effects models for each rating type. Table 16 shows the parameters for mixed effects models for each rating type. In order to best compare the effect of different parameters, here we show the parameter estimates and p-values for models with all the metrics that we used in the original planning behavior analysis from Section 4 of the main paper included as independent variables. Competence and directedness were both related to the proportion of interactions taken and percentage of stuck runs. This suggests that the semantic agents were rated as more competent and directed largely because they got stuck less and earned more reward. Additionally, competence and directedness were also related to percentage of successful runs. Since humans were successful more often, this partially explains why humans were rated as

more competent and directed. However, we also found a significant effect of being human above and beyond ability to succeed on this task, suggesting that humans exhibit other, holistic patterns of behavior that differentiate them from both agents. This pattern also emerged in humanness ratings to a lesser degree (with flipped signs for the "more human = less competent" participants), though this might just be a reflection of the correlation between humanness and competence.

References

Bentler, P. M., & Hu, L. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling*, *6*, 1–55.

Bloom, P. (1996). Intention, history, and artifact concepts. *Cognition*, *60*, 1–29.

Chaslot, G., Winands, H., & van den Herik, H. J. (2008). Parallel Monte-Carlo tree search. In *Proceedings of the international conference on computers and games* (pp. 60–71).

Chi, M., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, *5*(2), 121–152.

Daw, N., Gershman, S., Seymour, B., Dayan, P., & Dolan, R. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, *69*, 1204–1215.

Daw, N., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, *8*, 1704–1711.

Dennett, D. C. (1987). *The intentional stance*. MIT Press.

Diuk, C., Cohen, A., & Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on machine learning* (pp. 240–247).

Duncker, K. (1945). On problem solving. *vol. 58*, In *Psychological monographs*.

Gershman, S. (2021). *What makes us smart: The computational logic of human cognition*. Princeton University Press.

Gershman, S., & Niv, Y. (2015). Novelty and inductive generalization in human reinforcement learning. *Topics in Cognitive Science*, *7*, 391–415.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, *10*, 447–474.

Gopnik, A., Glymour, C., Sobel, D., Schulz, L., Kushnir, T., & Danks, D. (2004). A theory of causal learning in children: Causal maps and bayes nets. *Psychological Review*, *111*, 3–32.

Griffiths, T., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in Cognitive Sciences*, *14*, 357–364.

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. In *Proceedings of the 18th international joint conference on artificial intelligence* (pp. 1003–1010).

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., et al. (2018). DARLA: Improving zero-shot transfer in reinforcement learning. arXiv:1707.08475.

Kalish, M., Griffiths, T., & Lewandowsky, S. (2007). Iterated learning: Intergenerational knowledge transmission reveals inductive biases. *Psychonomic Bulletin & Review*, *14*, 288–294.

Kalish, M., Lewandowsky, S., & Kruschke, J. (2004). Population of linear experts: Knowledge partitioning and function learning. *Psychological Review*, *111*, 1072–1099.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., et al. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the 34th international conference on machine learning* (pp. 1809–1818). PMLR.

Kelemen, D., & Carey, S. (2006). The essence of artifacts: Developing the design stance. In E. Laurence, & S. Margolis (Eds.), *Creations of the mind: Theories of artifacts and their representation*. Oxford: Oxford University Press.

Kool, W., Cushman, F. A., & Gershman, S. J. (2018). Competition and cooperation between multiple reinforcement learning systems. In *Goal-directed decision making: Computations and neural circuits* (pp. 153–178). Elsevier.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, Article e253.

Lang, T., & Toussaint, M. (2010). Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, *39*, 1–49.

Litman, L., J. Robinson, J., & Abberbock, T. (2017). TurkPrime.com: A versatile crowdsourcing data acquisition platform for the behavioral sciences. *Behavior Research Methods*, *49*, 433–442.

Little, D., & Shiffrin, R. (2009). Simplicity bias in the estimation of causal functions. In *Proceedings of the Cognitive Science Society, vol. 31*.

Littman, M., Cassandra, A., & Kaelbling, L. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the twelfth international conference on machine learning* (pp. 362–370).

Liu, Y., Wu, Z., Ritchie, D., Freeman, W., Tenenbaum, J., & Wu, J. (2019). Learning to describe scenes with programs. In *International conference on learning representations*. URL https://openreview.net/pdf?id=SyNPk2R9K7.

Martin, J. J. (1967). *Bayesian decision problems and Markov chains*. Wiley.

Medin, D. L., Wattenmaker, W., & Hampson, S. (1987). Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology*, *19*, 242–279.

Nosofsky, R. (1987). Attention and learning processes in the identification and categorization of integral stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *13*, 87–108.

Pasula, H. M., Zettlemoyer, L. S., & Kaelbling, L. P. (2007). Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, *29*, 309–352.

Perez-Liebana, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S., Couetoux, A., et al. (2016). The 2014 general video game playing competition. In *IEEE transactions on computational intelligence and AI in games* (pp. 229–243).

Pouncy, T., Tsividis, P., & Gershman, S. (2021). What is the model in model-based planning? *Cognitive Science*, *45*.

Pu, Y., Ellis, K., Kryven, M., Tenenbaum, J., & Solar-Lezama, A. (2020). Program synthesis with pragmatic communication. In *Proceedings of the 34th conference on neural information processing systems*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Rosman, B., & Ramamoorthy, S. (2012). What good are actions? Accelerating learning using learned action priors. In *2012 IEEE international conference on development and learning and epigenetic robotics* (pp. 1–6). IEEE.

Rummery, G., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems*. Cambridge: Department of Engineering, University of Cambridge.

Saeedi, A., Kulkarni, T., Mansinghka, V., & Gershman, S. (2017). Variational particle approximations. *Journal of Machine Learning Research*, *18*, 1–29.

Schaul, T. (2013). A video game description language for model-based or interactive learning. In *Proceedings of the IEEE conference on computational intelligence in games*.

Schoenfeld, A., & Herman, D. (1982). Problem perception and knowledge structure in expert and novice mathematical problem solvers. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *8*, 484–494.

Scholz, J., Levihn, M., Isbell, C., & Wingate, D. (2014). A physics-based model prior for object-oriented MDPs. In *Proceedings of the 31st international conference on machine learning* (pp. 1089–1097). PMLR.

Schulz, E., Tenenbaum, J., Duvenaud, D., Speekenbrink, M., & Gershman, S. (2017). Compositional inductive biases in function learning. *Cognitive Psychology*, *99*, 44–79.

Sinz, F. H., Pitkow, X., Reimer, J., Bethge, M., & Tolias, A. S. (2019). Engineering a less artificial intelligence. *Neuron*, *103*, 967–979.

Tessler, M. H., Tsividis, P. A., Madeano, J., Harper, B., & Tenenbaum, J. B. (2021). Growing knowledge culturally across generations to solve novel, complex tasks. arXiv preprint arXiv:2107.13377.

Tomov, M., Schulz, E., & Gershman, S. (2020). Multi-task reinforcement learning in humans. *Nature Human Behavior*, *5*, 764–773.

Tsividis, P., Loula, J., Burga, J., Foss, N., Campero, A., Pouncy, T., et al. (2021). Human-level reinforcement learning through theory-based modeling, exploration, and planning. arXiv:1410.5401.

Tsividis, P., Pouncy, T., Xu, J. L., Tenenbaum, J. B., & Gershman, S. J. (2017). Human learning in atari. In *AAAI spring symposium series*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of probability and its applications, vol. 17* (pp. 264–280).

Weiser, M., & Shertz, J. (1983). Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies*, *19*, 391–398.

Wingate, D., Diuk, C., O'Donnell, T., Tenenbaum, J., & Gershman, S. (2013). *Compositional policy priors*: *MIT CSAIL Technical Report 2013-007*.

Wingate, D., Goodman, N. D., Roy, D. M., Kaelbling, L. P., & Tenenbaum, J. B. (2011). Bayesian policy search with policy priors. In *Twenty-second international joint conference on artificial intelligence*.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8*(7), 1341–1390.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., et al. (2018). Relational deep reinforcement learning. arXiv:1806.01830.