COGNITIVE SCIENCE
A Multidisciplinary Journal

Cognitive Science 45 (2021) e12928

ISSN: 1551-6709 online
DOI: 10.1111/cogs.12928

What Is the Model in Model-Based Planning?

Thomas Pouncy,[a] Pedro Tsividis,[b] Samuel J. Gershman[a,c]

[a]*Department of Psychology and Center for Brain Science, Harvard University*
[b]*Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology*
[c]*Center for Brains, Minds and Machines, Massachusetts Institute of Technology*

Received 15 August 2019; received in revised form 17 November 2020; accepted 17 November 2020

Abstract

Flexibility is one of the hallmarks of human problem-solving. In everyday life, people adapt to changes in common tasks with little to no additional training. Much of the existing work on flexibility in human problem-solving has focused on how people adapt to tasks in new domains by drawing on solutions from previously learned domains. In real-world tasks, however, humans must generalize across a wide range of within-domain variation. In this work we argue that representational abstraction plays an important role in such within-domain generalization. We then explore the nature of this representational abstraction in realistically complex tasks like video games by demonstrating how the same model-based planning framework produces distinct generalization behaviors under different classes of task representation. Finally, we compare the behavior of agents with these task representations to humans in a series of novel grid-based video game tasks. Our results provide evidence for the claim that within-domain flexibility in humans derives from task representations composed of propositional rules written in terms of objects and relational categories.

Keywords: Transfer learning; Reinforcement learning; Human cognition; Representation learning; Generalization; Artificial intelligence

1. Introduction

Human problem-solving is remarkably flexible. For example, although people typically learn to play chess on a standard board and with standard pieces, they can easily apply what they have learned to novel boards of different sizes, sets of pieces based on characters from beloved TV shows, or chess puzzles involving combinations of pieces that they have never seen before (e.g., 4 queens and 6 rooks). This flexibility extends beyond artificial tasks like chess to daily experiences in the real world. Driving in a new car, eating

Correspondence should be sent to Thomas Pouncy, Department of Psychology and Center for Brain Science, Harvard University, 52 Oxford St., Cambridge, MA 02138. E-mail: pouncy@g.harvard.edu

at a new restaurant, and using a new blender can all be accomplished with fairly minimal new learning.

Studies of problem-solving have shown that the degree of transfer between tasks depends on the choice of problem representation (Chi, Feltovich, & Glaser, 1981; Kaplan & Simon, 1990; Kotovsky, Hayes, & Simon, 1985). Once a sufficiently abstract representation (or schema) is acquired, it can be applied flexibly to different tasks and problem variants by simply plugging in the relevant parameters. By the same token, failures of transfer arise when people have failed to acquire a sufficiently abstract representation, or failed to recognize its conditions of application (e.g., Bassok & Holyoak, 1989; Gick & Holyoak, 1983). In some cases, the choice of problem representation can produce negative transfer, where performance on new problems is hindered by the overextension of representations that worked for previous problems (Duncker, 1945; Luchins, 1942).

Much of the work on flexibility in problem-solving has focused on cross-domain transfer (e.g., transfer between physics problems and algebra problems), but such problems are relatively infrequent in real life. More commonly, people are faced with the need to generalize across a wide range of variation within a domain. The visual system must be able to recognize objects from many different angles and distances. The speech recognition system must be able to recognize words spoken by many different speakers under many different acoustic conditions. The motor control system must be able to manipulate many different kinds of materials. These perceptual and motor invariances provide a well-studied example of flexible adaptation to variation within a domain. The purpose of the present paper is to deepen our understanding of the cognitive invariances that underpin flexibility in more abstract problem-solving domains. Following the logic of work on cross-domain transfer, and echoed by work on perceptual and motor invariance, we argue that the locus of flexibility lies in the problem representation: Generalization across variation is enabled by representational abstraction. Our goal is to characterize the nature of this abstraction in the domain of grid-based video games.

We chose to focus on grid-based video games because they strike a balance between realism and tractability. They incorporate some key aspects of real-world problems: They consist of objects and agents interacting spatially, with the dynamics of the environment following object-oriented relational rules. At the same time, these games are simple enough to be computationally tractable. This balance between realism and tractability has made video games an attractive target for research into computational models of problem-solving. An important observation from this work is that video games can be treated as sequential decision problems, for which a large body of reinforcement learning (RL) algorithms has been developed (Sutton & Barto, 1998). These RL algorithms have achieved dominance in many different kinds of games, far outstripping human performance when given enough training (Mnih et al., 2015; Silver et al., 2016, 2017). They have also matured as successful models of human behavior in simple sequential decision tasks with relatively small state spaces and short planning horizons (for a review, see Kool, Cushman, & Gershman, 2018). The application of these models to human behavior in more complex tasks like video games remains an active frontier (Anderson, Betts, Bothell, Hope, & Lebiere, 2019; Dubey, Agrawal, Pathak, Griffiths, & Efros, 2018;

Tsividis, 2019; Tsividis, Pouncy, Xu, Tenenbaum, & Gershman, 2017; van Opheusden & Ma, 2019).

Much of the empirical research on RL has focused on "model-free" algorithms that learn state-action values from trial-and-error experience. However, there is evidence that these algorithms are inadequate to explain the more structured patterns of decision-making present in video games and other complex tasks (Lake, Ullman, Tenenbaum, & Gershman, 2017; Tsividis et al., 2017). For this reason, we focus here on "model-based" algorithms that use a representation of the task to generate plans. As the central aim of this paper is on the relationship between task representation and within-domain generalization, we limit our focus to a single family of planning algorithms that have been shown to be capable of success in grid-based games. We simulate planning as a form of selective look-ahead search, which has been the basis of recent artificial intelligence (AI) game-playing success (e.g., Silver et al., 2016, 2017). The key question is how model-based planning (MBP) algorithms behave under different choices of model/task representation, and in particular which model choices lead to more human-like generalization behavior.

The rest of the paper is organized as follows. We first provide an overview of MBP, focusing on how different choices of model representation lead to different forms of flexibility. These models motivate a series of tasks that assay the corresponding forms of flexibility. We then report data from humans playing these tasks, and compare human game-play to game-play generated by planning agents equipped with different models.

2. Theoretical background

Generally speaking, MBP involves using a model of the task environment to simulate the outcome of various potential plans before selecting a course of action. In RL approaches to MBP, plans are selected based on their expected discounted future reward or value. The RL literature has proposed many different ways of estimating such value. However, as the current work is focused on how humans represent tasks rather than the algorithmic details of human planning, we will highlight the behavioral effects of distinct classes of task models within a single value estimation and planning framework. To ensure a fair comparison between these different model classes, we use a planning framework drawn from the family of Monte Carlo Tree Search (MCTS) approaches that have been successful in recent AI work on grid-based games (Silver et al., 2016, 2017). In particular, we use an approach outlined by Vodopivec, Samothrakis, and Ster (2017) that combines MCTS with Sarsa (λ) updates to produce value estimates, and a softmax function to select actions based on these estimates.

The first integral component of MBP is the model itself. The model stores whatever information the learning agent has about its current task. This information can take many forms, which we will explore in more detail in the next section, and allows the agent to make predictions about future task states based on currently observed states and proposed actions. In the context of RL algorithms a model consists of four key components: a

transition function (T), a reward function (R), a state space ($\mathbf{S}$), and an action space ($\mathbf{A}$). Given an initial state $s \in \mathbf{S}$ and a proposed action $a \in \mathbf{A}$, the transition function $T(s, a)$ returns a probability distribution over potential subsequent states $\mathbf{s}'$ that might occur as a result of taking action a from state s. Similarly, $R(s)$ returns a distribution over potential rewards that might be obtained from reaching state s. Together these two functions can be used to simulate the potential outcomes of different plans during planning.

One of the difficulties of planning in sequential decision tasks is that many environments feature a large number of states or actions, of which only a small number lead to reward. These sparse rewards require more exploration to discover. RL techniques for value estimation, on the other hand, often only become accurate after many observations of the same transition. MCTS balances this trade-off between exploration and accuracy by selecting actions based on a combination of their estimated value and the uncertainty around that estimate (Kocsis & Szepesvári, 2006). In particular, MCTS calculates statistical confidence bounds around the estimated value of each available action as follows:

$$v_a \pm \sqrt{\frac{2 \times \ln(n_a)}{N_s}},$$

where v_a represents the current value estimate of action a from past simulations, n_a represents the number of times action a has been simulated from the current state, and N_s represents the total number of simulations run. Potential action trajectories are generated by selecting actions with the highest upper confidence bound. Thus initially promising actions are selected more frequently, which lowers their upper confidence bound. The greater uncertainty around less frequently simulated actions eventually causes them to be selected instead, driving the algorithm to alternate between providing more accurate value estimates of previously rewarding options and exploring less-visited options (for a more in-depth review of MCTS and its variants, see Vodopivec et al., 2017).

Estimating the value of a particular state/action pair within a simulated episode involves identifying the causal link between that pair and any future earned reward. While this credit assignment problem is notoriously difficult to solve, the RL literature has proposed a number of approaches for approximating a solution. In this work, we use the Sarsa (λ) algorithm developed by Rummery and Niranjan (1994), in which the value of each state/action pair is estimated incrementally according to:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \times [R(s_{t+1}) + \gamma \times Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)],$$

where $Q(s_t, a_t)$ represents the current estimated value of taking action a_t from state s_t based on past simulations, $R(s_{t+1})$ reflects any immediate simulated reward earned as a result of reaching state s_{t+1}, and $Q(s_{t+1}, a_{t+1})$ represents the current estimated value of the next action in the simulated action sequence (for a more in-depth review of temporal difference learning and its implementations, see Sutton & Barto, 1998). Thus MCTS and Sarsa (λ) work together to provide balanced simulations of potential action trajectories and estimate the value of these plans.

Finally, we used a softmax function to convert value estimates to selected actions. Given some assumptions, it has been shown that deterministically selecting the action with the highest value estimate at a given point in time leads to more accurate value estimates asymptotically (Sutton & Barto, 1998). However, empirical data in humans have demonstrated that human decision-making is often better modeled as a stochastic rather than deterministic function of the value associated with a particular choice (Luce, 1959). To capture this empirical phenomenon, we use the following softmax function to select actions:

$$P(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}}.$$

Here, $P(a|s)$ represents the probability of selecting a particular action a in state s, and τ is a temperature parameter that controls the stochasticity of the choice. For more detailed pseudo-code and simulation details pertaining to the value estimation and action selection processes, see Appendix C in the Supporting Information.

3. Task representations

The model in MBP can take many forms. Classical RL approaches to MBP have often attempted to model human problem-solving using tabular Markov decision processes (MDPs) to represent tasks. A tabular MDP models task dynamics by mapping each state of the world s and available action a onto compressed representations $C_S(s)$ and $C_A(a)$. This pair of compressed representations $(C_S(s), C_A(a))$ is then used as keys in a lookup table K. Each entry $K[C_S(s), C_A(a)]$ in this table corresponds to a probability distribution over the possible states $\mathbf{s}'$ (with compressed representations $\mathbf{C_s}(\mathbf{s}')$) that one might transition to as a result of taking action a from state s.[1] These compressed state and action representations, when combined with the transition function T and reward function R, represent an agent's overall understanding of the task. We will refer to this combination of C_S, C_A, T, and R as an agent's "task representation" for brevity. Under this formalism, any transition dynamics that have been learned for state s and action a would naturally generalize to any other states $\mathbf{s}^*$ and $\mathbf{a}^*$, where $\mathbf{s}^*$ and $\mathbf{a}^*$ are the sets of states and actions that compress to the same lookup key as s and a (i.e., $C_S(s) = C_S(s^*) \forall s^* \in \mathbf{s}^*$ and $C_A(a) = C_A(a^*) \forall a^* \in \mathbf{a}^*$). This suggests that different task representations can produce different behavioral signatures even within the same planning framework, as different representations will allow agents to generalize to different sets of other states. Drawing on this observation, we propose that generalization across within-domain variation arises as a behavioral signature of the particular task representation that humans use for MBP.

Recent work on representation learning has proposed a number of ways of representing complex tasks. Ponsen, Taylor, and Tuyls (2010) suggest that these task representations can be broadly categorized as representations built from useful object features, useful object categories, or some combination thereof. Bengio, Courville, and Vincent (2013) further discuss representation learning as a form of feature extraction, wherein primitive

representational elements are grouped together or filtered to provide more useful state representations. In the context of complex sequential decision tasks that are performed on a computer, machine learning approaches have often used the pixels on a screen and buttons on a computer as representational primitives. In contrast, models of visual cognition in humans are often grounded in the receptive fields of neurons in early visual pathways, or low-level visual features like edge-detectors and Gabor filters. However, while these kinds of low-level representational primitives are popular in models of visual cognition and machine learning, there is ample evidence that more complex reasoning and problem-solving processes in humans are grounded in conceptual primitives like objects and relations (Halford, Wilson, & Phillips, 2010; Spelke, 2003; Spelke & Kinzler, 2007). Following this work, a wide array of recent work in AI and computational cognitive modeling has already begun to approach state representation learning as a problem of identifying useful groupings of objects based on features and relational properties (Diuk, Cohen, & Littman, 2008; Garnelo & Shanahan, 2019; Piantadosi, 2011; Santoro et al., 2017; Zhu, Huang, & Zhang, 2018). Integrating biases for object-oriented (Higgins et al., 2018) or relational (Zambaldi et al., 2018) representations has produced zero-shot, within-domain transfer in a number of simple tasks and state-of-the-art performance on more complex tasks like StarCraft. The success of these novel AI approaches across many variants of similar tasks suggests that MBP with task representations made up of visible object features or relational properties might be enough to account for human generalization in domains like grid-based games.

On the other hand, a subset of recent AI work has begun to move away from feature vector or relational representations and back toward more abstract rule-based representations (Lang, Toussaint, & Kersting, 2012; Zettlemoyer, Pasula, & Kaelbling, 2005). Drawing on early rule-based AI systems, these rule-based approaches focus on identifying if-then statements capable of describing the interaction patterns of specific sets of objects or relational categories. For example, where an object-feature-based approach might represent the game of chess as a series of distinct states each represented by the position and color of the pieces on a chess board, a rule-based approach might instead represent chess as a series of rules like "if the piece is a white pawn, then it can move one square up" and "if a piece lands on a square containing another piece, then the second piece is removed from the board." These rules can then be combined to represent the large space of possible states of a chess game in a more compact fashion.

Although feature vector, relational category, and rule-based representations can all be used to represent the same tasks, the varying levels of abstraction involved in each representation predict distinct patterns of behavior. Take, for example, a task in which there are doors that visibly lead to treasures or traps. When exposed to instances of this task in which treasures are always blocked by green doors while traps are always blocked by red doors, an agent that internally represents task states as sets of object categories may eventually learn that green doors lead to states associated with reward. If the agent was then exposed to task instances where green doors lead to traps and red doors to treasures, such an agent would require some additional training experience to reverse its previous associations. In contrast, a rule-based agent might learn several different rules, including that

doors can be opened by touching them, picking up treasures leads to reward, and touching traps leads to punishment. By applying all three rules to the same state, the rule-based agent would similarly conclude that it is desirable to open green doors when they block treasures. However, the exact same rule-based representation would allow the agent to conclude that it should not open green doors when they lead to traps without the need for any additional training. When exactly an agent can succeed in a previously learned task without the need for additional training is the key distinguishing factor between these three different task representations. We will refer to this ability to succeed at novel task variants without additional training as "flexibility" for brevity. In the next section, we present three task representations corresponding to the output of feature vector, relational category, and rule-based approaches to representation learning, as well as a baseline task representation for comparison. We then present a series of tasks designed to assay the flexibility of these four representations. Finally we compare the performance of these four representations to human behavior on the same tasks.

3.1. Baseline representation

Our first three task representations used a tabular MDP for a transition function. As specified in Section 3, a tabular MDP maps primitive state/action representations to compressed task representations. These compressed representations are then used to look up a probability distribution specifying likely transitions to other states. For all four task representations, our primitive state representations consisted of the set of all objects on screen in a given state. Each object had color, shape, and position features as well as a latent variable for whether or not that object was currently being held in the player's inventory. Our primitive action representations mapped to the four movement directions available in each task (i.e., up, down, left, right). To evaluate the performance of an agent without any additional representational abstractions, we included a baseline agent whose compressed state/action representations were the same as the game's primitive state/action representations. Finally, we used a simple reward function $R(s) = p - w_c \times c$, where p represented the number of points earned up to this time step, c represented the number of actions taken before arriving in this time step, and w_c represented a weight parameter for controlling the cost of additional choices relative to earning additional points. We refer this model as the baseline model. Table 1 provides an example of how this representation would capture the state/action transition shown in Fig. 1.

One additional difficulty of using tabular MDPs as a transition function is that the lookup table at the heart of the MDP can become quite large, even for very simple tasks. To ensure a fair comparison between agents with different task representations, we wanted each agent's transition function to be as close as possible to what the agent would have obtained during training under an ideal representation learning process. It would have been impractical to try and fill in the hundreds or thousands of observed state transitions that an agent would have stored during training entirely by hand. Thus, we instead used a simple update process to fill in the lookup tables used by each tabular task representation. This update process is described in more detail in Appendix A.

Table 1
Summary of key model components for the four models considered in the paper. "State A" and "Action A" show how the agent would represent the state/action shown in Fig. 1A, while "State B" and "Action B" show how the agent would represent the state/action shown in Fig. 1B. Features in vector representation are "color," "shape," "x-position," and "y-position." In Fig. 1A, color is informative, while in Fig. 1B, shape and color are jointly informative

Model Class	Storage Format	State A	Action A	State B	Action B
Baseline	Tabular MDP	{ ("pur.", "cir.", 1, 2), ("pur.", "cir.", 2, 3), ("gr.", "cir.", 2, 2), ("yel.", "cir.", 3, 2), ("yel.", "cir.", 2, 1) }	"right"	{ ("pur.", "cir.", 1, 2), ("pur.", "cir.", 2, 3), ("gr.", "sq.", 2, 2), ("pur.", "sq.", 3, 2), ("pur.", "sq.", 2, 1) }	"right"
Inf.-obj.-features	Tabular MDP	{ ("pur."), ("gr."), ("yel.") }	("yel.") + BFS navigator	{ ("pur.", "cir."), ("gr.", "sq."), ("pur.", "sq.") }	("pur.", "sq.") + BFS navigator
Rel.-categories	Tabular MDP	{ ("treasure"), ("avatar"), ("trap") }	("treasure") + BFS navigator	{ ("treasure"), ("avatar"), ("trap") }	("treasure") + BFS navigator
Rule-based	Prop. rules + logic engine	{ ("pur.", "cir.", 1, 2), ("pur.", "cir.", 2, 3), ("gr.", "cir.", 2, 2), ("yel.", "cir.", 1, 3), ("pur.", "cir.", 3, 2) }	("pur.", "cir.", 2, 1) + BFS navigator	{ ("pur.", "cir.", 3, 2), ("pur.", "cir.", 2, 3), ("gr.", "sq.", 2, 2), ("pur.", "sq.", 3, 2), ("pur.", "sq.", 2, 1) }	("pur.", "sq.", 3, 2) + BFS navigator

3.2. Informative-object-features representation

We designed our second model to represent the output of an object-feature-based representation learning process. Previous work has suggested that learning useful state/action representations involves identifying combinations of key object features that are ultimately predictive of object values (Bengio et al., 2013; Ponsen et al., 2010). This feature extraction process provides a powerful representation learning approach where classical feature learning or dimension reduction techniques are applied to sets of vectors encoding the features of all objects in a task (for a review of a variety of such techniques in relation to representation learning, see Zhong, Wang, Ling, & Dong, 2016). The expected result of such a learning process would be a state representation where all the informative features have been combined into novel aggregate features that capture some underlying element of the task environment. Several recent approaches have looked at applying classical RL to the ideal output of these kinds of feature extraction processes, and found the

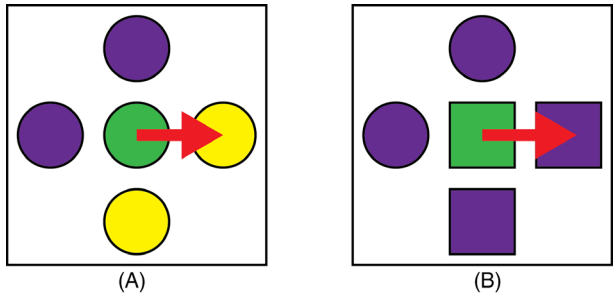

Fig. 1. Two examples of action selection during simple tasks. Red arrows indicate the action selected by the player on this game screen. The player controls the green shape. (A) The yellow shapes are rewarding, while the purple shapes are punishing. (B) Objects that are of the same shape as the avatar are rewarding, while objects that are of a different shape are punishing.

combination to be successful at a variety of tasks (Mohan & Laird, 2011; Raffin et al., 2019). To replicate this within the context of tabular MBP, we implemented an agent that was given ground truth access to which subset of object features were informative for each task. The set of informative feature vectors for all objects in a scene was then used as the compressed state representation $C_S(s)$ for this model.

Recent work on hierarchical RL (HRL) has demonstrated a key relationship between an agent's state and action representations. This work highlights the fact that many sequential decision problems contain repeated sets of actions that are related to solving the same subgoals. At a high level, many of the computational approaches for learning which subgoals are useful for a given task involve identifying commonly traversed task states that serve as bottlenecks between large sets of similar states (Hengst, 2012; Tomov, Yagati, Kumar, Yang, & Gershman, 2020). For example, a doorway between two rooms represents a bottleneck linking any state in the first room to any state in the second room. Thus navigating to that doorway is likely to be a useful subgoal from within either room. As the set of bottleneck states is dependent on the particular state representation used, this suggests that useful action representations will be similar in form to useful state representations for a given task.

Following this intuition, we provided our second model with more abstract action representations to better match its state representations. Rather than up, down, left, or right directions as an action space, our second model instead treated all of the objects in a state as potential subgoals to select. We used a breadth first search (BFS) algorithm to map each target object to a set of directional actions for reaching that target. This mapping can be thought of as the output of an ideal HRL process. Finally, this model used the same reward function as the baseline model. However, the number of actions taken (c) in the reward function for this model referred to the number of subgoals that were selected, rather than the number of individual directional actions taken. We refer to this model as the informative-object-features model. Table 1 provides an example of how this representation would capture the state/action transition shown in Fig. 1.

3.3. Relational-categories representation

We designed our third model to reflect recent efforts to incorporate relational reasoning into representation learning. One of the challenges of classical feature learning is that it often struggles to extract useful information about the relationships between features. Recent work in AI has attempted to address this by providing explicit relational primitives to representation learning architectures grounded in artificial neural networks (Santoro et al., 2017). While the precise representations learned by this class of algorithm are often difficult to extract and apply to humans, work on cognitive modeling of human category learning has explored the relational information learned by humans more directly. Piantadosi (2011) suggests that humans naturally group objects into categories based on Boolean functions composed of relational predicates. For example, rather than learning categories like "key," "door," and "decoy key" by mapping them to particular features like "shape" or "color" that may not always be informative, humans can also map these categories onto more abstract descriptions like "thing that is the same color as a door," "thing that is the same color as a key," and "thing that is the same shape as a key but a different color from every door." By grounding these descriptions in a formal logical language, it becomes possible to formalize cognitively plausible learning processes for discovering these categories. Piantadosi (2011) provides a more in-depth description of both the propositional language involved in these categories and potential learning mechanisms for inferring these categories from data. In order to ascertain the behavioral effect of using such relational, propositional representations within a tabular MBP context, our third model internally represented each state as the set of relational propositions corresponding to each object's category assignment within a task (e.g., "door" objects are the things "that are the same color as keys," etc.). The relational categories used by this model were the ground truth object categories that made up each task. These ground truth categories were explicitly provided to the model. As with the second model, this relational-categories representation was given abstract action representations to match its state representation. Thus, this model selected relational categories as subgoals and used the same BFS navigator to return the shortest path from the player's current position to an object of that category. We refer to this model as the relational-categories model. Table 1 provides an example of how this representation would capture the state/action transition shown in Fig. 1.

3.4. Rule-based representation

Finally, our fourth model draws on recent work in AI on factored, rule-based representations within MBP. A number of different structures for representing and learning rules have been proposed, including object-oriented RL (Diuk et al., 2008), relational RL (Lang et al., 2012), and noisy deictic rule learning (Zettlemoyer et al., 2005). All of these approaches can be broadly summarized as learning a series of statements of the form "preconditions + action → outcome." In this format, actions and preconditions reflect a series of propositions over objects and relational categories while outcomes reflect transformations that can be applied to sets of objects (for an in-depth formalization of this rule

structure, see Zettlemoyer et al., 2005). Intuitively, this provides a precise method for formalizing abstract rules like "touching a door while there is a key that is the same color as the door in the inventory causes the door to disappear" into a representation like the following:

$$r_i : \lambda < actor, target, inventory > .$$
$$\texttt{Touch}(actor, target) \wedge$$
$$\texttt{isDoor}(target) \wedge$$
$$\texttt{isKey}(inventory) \wedge$$
$$\texttt{matches}(target, inventory)$$
$$\rightarrow$$
$$\texttt{remove}(target) \wedge \texttt{remove}(key)$$

where `isDoor()` and `isKey()` are the same relational category functions used by the relational-categories model. The task rules used by this agent were hand-coded to best represent the ground truth transition dynamics of the actual tasks in the rule language used by the agent.

It is important to note here, that while it is possible to use rule-based representations to generate tabular MDPs, doing so would require complex processes for identifying ideal state representations for each set of task rules. Thus, in practice it is often easier and more computationally efficient to combine these rules with a simple logic engine. By identifying which rules would hold in a given state and then evaluating the cumulative effect of applying those rules, this logic engine can provide the same transition function information as the lookup table in a tabular MDP while taking up significantly less space. Thus, for the current work we provided our rule-based agent with such a logic engine for the sake of tractability and efficiency.

Since the rule-based transition function already handles compressing the internal representation of the task dynamics by mapping them to compact rules, this model type does not also require a separate, compressed state representation C_S. Thus, this agent simply used the game's primitive state representation. However, for the purposes of comparing the behavior of this rule-based task representation to the informative-object-features and relational-categories representations, we also provided this representation with an object-based action space. This model selected individual objects as targets and used the same BFS navigator to convert these subgoals to directional actions. We will refer to this model type as the rule-based task representation. Table 1 provides a summary of the key components of these four agent types, as well as examples of the state and action representations used by each model. Appendix A provides additional details about incorporating each of these four representations into the MBP framework described in Section 2.

4. Task design

In this section, we present four tasks designed to compare the within-domain flexibility of our four task representations to human performance. While humans can flexibly adapt to many different kinds of within-domain variation, we will focus on three prominent examples of such flexibility that best differentiate the task representations mentioned in Section 3: changes in the number and position of objects, changes in the features of objects, and changes in the composition of object functions within a task. Each of these forms of cognitive invariance is common in the sequential decision domains that humans face in everyday life.

To demonstrate how humans can succeed at a previously learned task in spite of variations along these three dimensions, we designed four simple games. For each kind of flexibility that we wished to elicit, we designed a game with consistent rules and then generated many instances of that game that varied along the desired dimension. To avoid any potential transfer between tasks, each game was played in isolation.

4.1. The object number variation task

Our first task varied the number and position of objects across task instances. In real-world tasks like crossing the street or navigating the grocery store, humans are often agnostic to the exact layout of objects they face. Humans can cross the street safely regardless of the number of cars parked along the side of the street, and they can navigate the grocery store no matter how many cans there are on the shelves. The object number variation task (ONVT) captured this phenomenon with a simple game involving treasures, traps, walls, and a goal object. Two example instances of this game are shown in Fig. 2. In both of these instances, the player controlled the red avatar with the red border and moved it around the screen. Touching the green shapes with the avatar earned the player three points, while touching the purple shapes lost the player two points. The avatar could not move through the cyan shapes, and touching the other red shape ended the game. Individual instances of this task contained varying numbers of traps, treasures, and walls. Appendix B provides additional information about how these instances were generated.

4.2. The simple object feature variation task

Our second task varied the identity of object features across task instances. In real-world scenarios, humans are capable of representing the fact that certain object features are irrelevant to task performance. For example, chess players know that whether chess pieces are black and white, red and green, or gold and silver is irrelevant as long as the pieces come in two different colors. Our object feature variation task (OFVT) captured this phenomenon using a simple game involving keys and doors in addition to the treasures and walls from the ONVT. Two examples of this task can be seen in Fig. 3. In this task the player controlled the avatar shape bounded by the red box. At any point, the player could finish the game by touching the goal shape that looked visually identical to the avatar. Each example of this game also included multiple walls (the pink, clipped

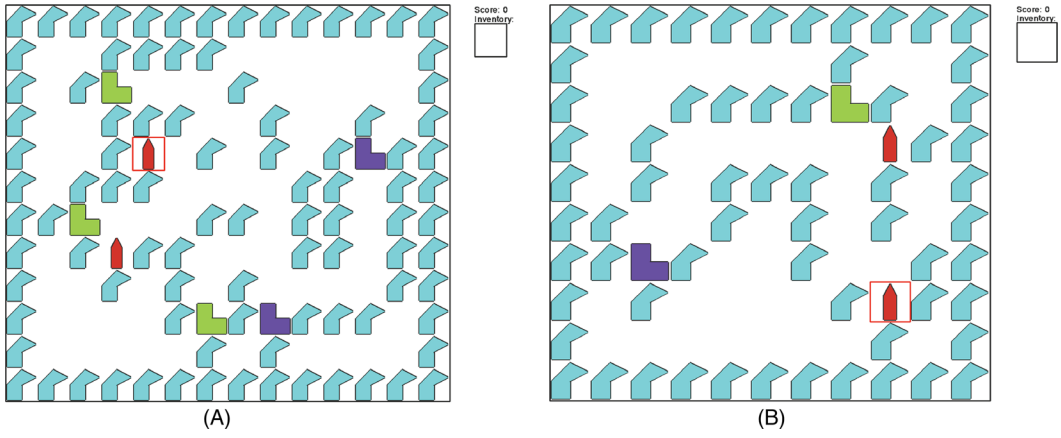

Fig. 2. Two examples of the object number variation task. In these examples, the agent controlled the red shape surrounded by a red box. The cyan shapes acted as walls. Touching the green shapes earned the agent 3 points, while touching the purple shapes lost the agent 2 points. Touching the other red shape ended the game. (A) shows an example with three rewarding objects and two punishing objects, while (B) shows a variation with only one of each.

squares in Fig. 3A) and treasures (the blue shapes in Fig. 3A) just like in the ONVT. However, this game also introduced impassable "door" objects (the green pencil shape in Fig. 3A), as well as "key" objects (the red, green, and cyan shapes in Fig. 3A). Touching the key objects caused them to move into the player's inventory (visible along the right side of the screen). If the inventory was already full, the new key was swapped with the current inventory content. This meant that the player could only carry one item at a time. Doors disappeared if and only if the player touched them while a key of the same color was in the inventory slot. Each example also included at least one decoy key that did not match the color of any door in order to make it less likely that the player could open a door without understanding how the keys worked. The color of each type of object varied from instance to instance of this task; thus, walls might be yellow in one instance and green in another. Appendix B provides additional details about the generative process used to create instances of this task. We refer to this task as the simple object feature variation task (SOFVT).

4.3. The abstract object feature variation task

In the SOFVT, at least one feature (shape) remained informative from instance to instance. However, previous work has demonstrated that humans are also capable of learning completely abstract categories that are defined only by their relation to other categories (Piantadosi, 2011). In order to elicit this phenomenon, we created a second variant of the OFVT that we refer to as the abstract object feature variation task (AOFVT). As shown in Fig. 4 both the color and the shape associated with each object type varied from instance to instance of this task.

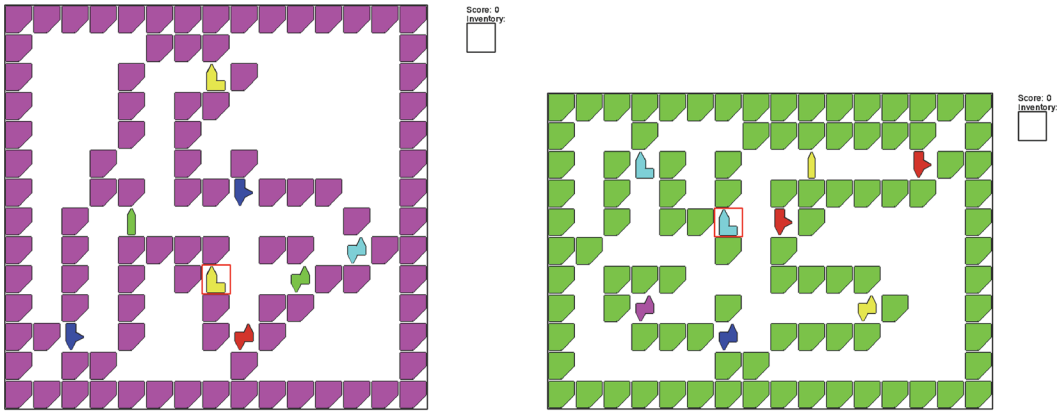

Fig. 3. Two examples of the simple object feature variation task. In these examples, the agent controlled the object surrounded by a red box. The clipped square shapes acted as walls. Touching lightning bolt shapes added them to the agent's inventory (shown on the right-hand side of each example). The pencil-shaped objects acted as doors. Each door would only open if touched while a key of the same color was in the inventory. The remaining objects were treasures, which awarded 3 points when picked up. (A) shows one example of this task using a yellow avatar and exits with pink walls, while (B) shows a variant of the same task using a blue avatar and exit with green walls.

4.4. The object composition variation task

Finally, our last task varied the composition of object functions across task instances. In the real world, humans are capable of representing information about how an object functions independently of the context in which that object was first observed. For example, while one might initially observe a hammer being used to secure nails, one is capable

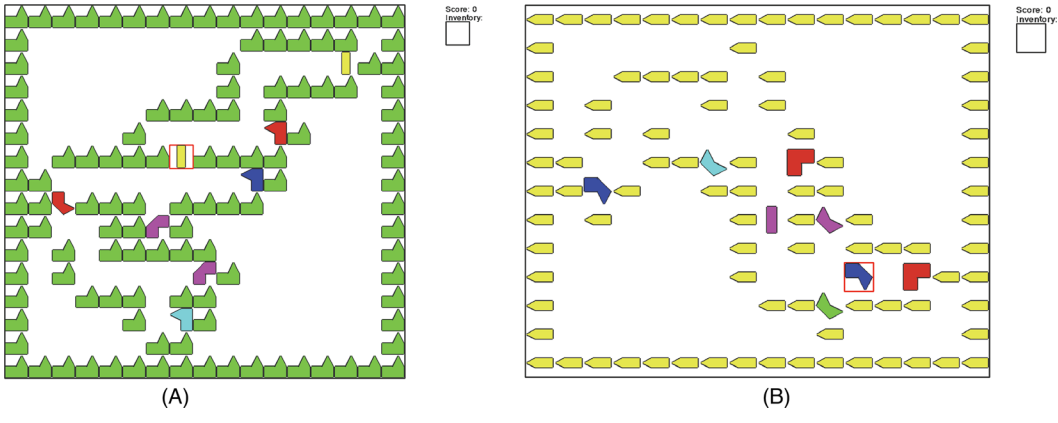

Fig. 4. Two examples of the abstract object feature variation task. In these examples the agent controlled the shape in the red box. The green objects in (A) and yellow objects in (B) acted as walls. The pink objects in (A) and red objects in (B) acted as treasures that awarded 3 points when touched. The remaining shapes acted as keys and doors as in the simple robust relations task.

of imagining using the same hammer to apply force to other objects as well. The object composition variation task (OCVT) captured this phenomenon using a simple game involving conveyor belts and teleporters as well as the treasures, traps, and walls from the ONVT. Several examples of this task can be seen in Fig. 5. In this task, players controlled the avatar (purple shape with red border in Fig. 5) and navigated to the identical goal object while picking up treasures (cyan shapes in Fig. 5) and avoiding traps (red shapes in Fig. 5). However, this game also introduced teleporter objects, which consisted of two sub-kinds: senders and receivers (the bent diamond and horizontal bar shapes, respectively, in Fig. 5). Touching a sender instantaneously transported the avatar to the corresponding receiver, as indicated by matching color. These teleporters were equally likely to lead to treasures as traps; thus, success on the task required correctly predicting the destination of each teleporter. The first block of the task introduced players to this subset of elements, shown in Fig. 5, so that they were not overwhelmed with too many elements at once.

In the second block of this task, shown in Fig. 5B, we introduced conveyor belt objects (the dark blue lines and l-shapes in Fig. 5B). Touching a conveyor belt caused the avatar to move along a path that was the same shape as the conveyor belt object itself. The avatar would then automatically interact with any objects encountered along the way. In the second block of this task, conveyor belts led to either treasures or traps with equal probability and each conveyor belt was preceded by a receiver and followed by a corresponding sender. This meant that touching a conveyor belt would force the player to interact with either a treasure or a trap before being teleported back to the space before the conveyor belt. Similar to the first block, success in this block required correctly predicting which conveyor belts would lead to treasures and which would lead to traps.

It would still be possible to succeed on this task without any understanding of how a conveyor belt works by simply implementing the heuristic "interact with conveyor belts when they are near treasures and not when they are near traps." Thus, in the third block, shown in Fig. 5C, we introduced local distractor objects to ensure that successful agents had to properly model the true mechanics of the task. Here conveyor belts worked exactly as before, only now we placed several alternate paths after each belt. One path was always the path that the conveyor belt actually followed, leading with equal probability to either a treasure or a trap as before. The new, alternate path always led to the opposite outcome as the true path. This ensured that each conveyor belt was equidistant from one treasure and one trap, making it difficult to predict the outcome of interacting with a belt based solely on nearby objects.

Finally, in order to fully test participants' understanding of conveyor belt function independent of local context, we added a fourth block of this task that involved chaining multiple conveyor belts together so that each belt led to another belt. An example of the fourth block of this task is shown in Fig. 5D. Each example of this task involved novel combinations of the three kinds of conveyor belts, treasures, and traps. We refer to this task as the OCVT, and when we mention results from this task we will only be discussing results from the fourth, most challenging block.

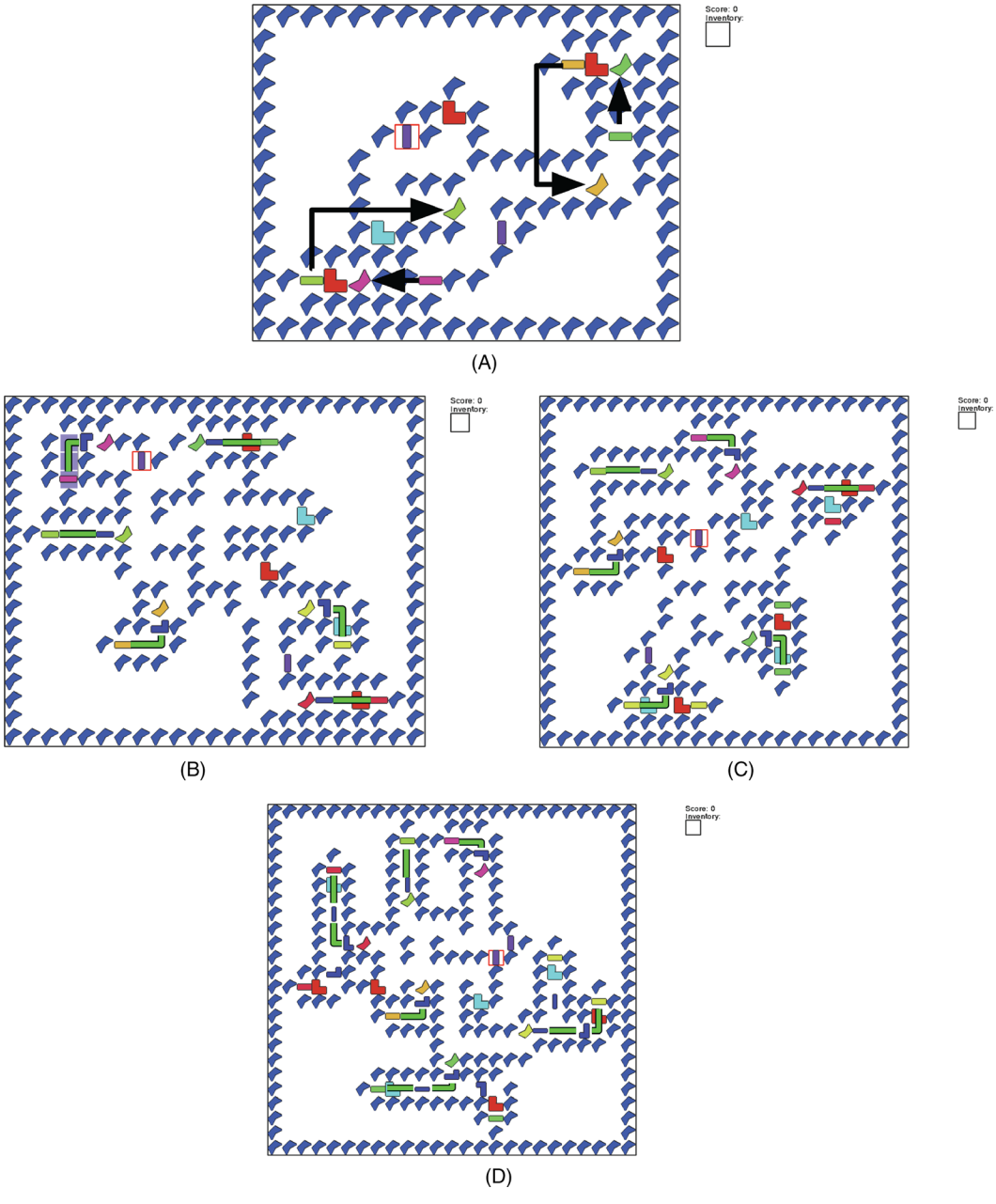

Fig. 5. Examples from the four stages of the object composition variation task. Here the agent controlled the shape surrounded by a red box. The blue objects acted as walls. The red shapes took away 2 points when touched, while the cyan shapes awarded 3 points. The remaining objects acted as teleporters or conveyor belts, as described in the main text. Black arrows represent where teleporters sent the agent, and green lines indicate the path conveyor belts followed. Neither were visible to the participant during play. (A) shows an example from the first block of this task that uses teleporters without conveyor belts. (B) shows an example from the second block using conveyor belts with a single possible path. (C) shows an example from the third block with conveyor belts leading to multiple potential paths, and (D) shows an example from the fourth block using conveyor belts linked together in sequence.

5. Experimental methods

5.1. Participants

To establish an empirical baseline for human flexibility to within-domain variation along these dimensions, we collected data from 165 human participants using Amazon Mechanical Turk (AMT). In order to ensure participant quality, we required all of our participants to have a previous approval rating of 90% or higher. Additionally, since many workers on AMT request many tasks at once, we included a comprehension check to filter out participants who were not fully focused on our task. The comprehension task consisted of providing a written description of the rules of the task after all task instances had been completed. We filtered participants based on a number of criteria. First, we removed the 22 participants who failed to provide any rule descriptions at all (e.g., "Good game, very interesting" or "I like this kind of task"), as these participants gave no strong indication that they had paid attention to the task instructions. We filtered out an additional 56 participants who either failed to describe key rules or who described one or more rules incorrectly. The rest of this analysis concerns the remaining 84 participants who correctly described the rules of their task. This group contained 20 participants who completed the ONVT, 21 who completed the SOFVT, 22 who completed the AOFVT, and 21 who completed the OCVT. A subsequent analysis suggested that our comprehension check may have been overly stringent, as the majority of the participants who failed the check still had qualitatively similar performance to those who passed. However, in the interest of following our initial design, we restricted the analyses presented here to the participants who passed the comprehension check.

5.2. Design and procedure

To avoid any potential transfer between tasks, we used a between-subjects design where each participant was randomly assigned to one of the four tasks described in Section 4. In order to avoid biasing participants toward the rules used in the rule-based representation, we provided no instructions about the rules of each task beyond the controls for interacting with the game. Participants were told that they would see several instances of a game and that their goal was to earn as many points as possible in that game by using the mouse to click on different objects. Clicking on an object would cause the player's avatar to begin navigating to the target object, and participants could click on a new object at any point to re-route the avatar to the new target.[2] Participants played through 10 instances of their assigned task, except for the participants who were assigned to the OCVT, who played through only 5 instances of each of the four blocks to avoid fatigue. Object positions and appearances for each instance were generated at random for each participant according to the constraints of each task. Assigning different object appearances to each participant allowed us to control for any latent associations that participants might have had coming into the task (e.g., "red objects are bad" or "green objects are good"). Each task instance was unique along its key dimension of variation. Thus, instances of the ONVT were unique in the number of each object type and position

of each individual object, while instances of the OFVT were unique in the color or shape assigned to each object type. Appendix B provides additional details about the position and appearance randomization process. After each instance, participants were provided with feedback about how many points they earned, as well as how many points they could have earned.

For each task we tracked the set of target objects that players chose to interact with as well as the player's final score. Since each task instance involved interacting with a different number of rewarding or punishing objects, the total number of points that one could earn varied from instance to instance. To address this, we tracked the player's score as a percentage of the total points they could have earned on each instance. Additionally, we tracked the total quantity of interactions the player had with each type of object (e.g., treasure, trap, wall, etc.) in order to get a more fine-grained picture of participant behavior.

Finally, as the focus of this project is on generalization within previously learned domains, we needed a way to identify when participants had fully learned a task. Based on initial observation of human learning curves during our pilot, we found that once participants reached their peak performance on a task, they tended to maintain that performance across subsequent instances. Drawing on this observation, we marked all instances up to and including the first instance of perfect performance as training instances and all instances after that as evaluation instances. When comparing model behavior to human behavior, we used only these evaluation instances. As humans learned most of these tasks within two to three instances, 1,151 of the 1,480 total task instances played met the criteria for an evaluation instance. Only a few participants had no instances that met the criteria for an evaluation instance. Removing these participants, we were left with 20 participants each for the ONVT, SOFVT, and OCVT, and 21 participants for the AOFVT.

5.3. Model fitting

To evaluate the capabilities of each task representation under optimal conditions relative to human behavior, we first fit the parameters of a model with each representation to maximize overall task performance. To match sample sizes between human participants and computational models, we ran one instance of each model class for each human participant collected. Thus, we ran 81 instances of agents with the baseline, informative-object-features, relational-categories, and rule-based representations. Each agent was shown the exact same evaluation instances as their corresponding human participant and had six parameters (N_s, α, γ, λ, τ, and w_c); see Section 3 and Appendix C for more details. To find the optimal performance parameters for each model class, we did a grid search over all six parameters with the goal of maximizing the quantity $Z = \rho_p - s_p$, where ρ_p was the percentage of total possible points earned and s_p was the percentage of total allowed steps taken, averaged across all four tasks. As some parameters caused agents to select the same action or target an infinite number of times, all agents were limited to a maximum of 325 directional actions or 15 target object selections in order to obtain parameter fits in a reasonable amount of time. Adding a penalty for parameters that caused agents

to take more steps was done as an additional effort to speed the parameter fitting process. A comparison of parameters fit without this step penalty on a subset of agents showed no substantive difference in the pattern of results described below. For the baseline, informative-object-features, and relational-categories representations, each individual agent was trained to asymptotic performance in order to ensure that the agent had the best available tabular representation of that form. We collected each agent's average evaluation instance performance only after the agent reached asymptotic training performance. Additionally, we included a random agent who was equally likely to select any of the four directional keys as a non-model-based comparison.

For a more quantitative evaluation of overall model fit, we also evaluated the same agents with parameters fit to maximize the log-likelihood (LL) of the human data under a given model class. More specifically, each model class produced a probability distribution over the actions available from any given state. We used this distribution to evaluate the likelihood of the set of actions generated by humans under each model class. As our tasks were Markovian and our agents did not cache values or simulations between decision points, the probability of the human behavioral data under each model $P(D|M)$ could be specified as follows:

$$P(D|M) = \prod_{i=1}^{|D|} \prod_{a,s \in d_i} P(a|s,M,\theta_i),$$

where i represents each human participant in our data set D, d_i is the data for each participant, and θ_i are the model parameters that best fit that participant. Since the transition dynamics and simulation algorithm from each model were used to estimate the potential value of all available actions from a given state, and these value estimates were used to select actions under a softmax policy, we can express the above likelihood as:

$$P(a|s,M,\theta_i) = \int_{V(s)} P(a|s,V(s))P(V(s)|M,\theta_i)dV(s).$$

The first quantity, $P(a|s, V(s))$, was provided explicitly by the softmax policy selection function. The second quantity is more difficult to calculate analytically. However, further inspection of samples from the $P(V(s)|M, \theta_i)$ distribution showed that the variance in value estimates of particular actions within a given state was much lower than the variance in value estimates between actions in the same state. As the probability of action selection given the value estimates largely depends on the relative magnitudes of the values of actions within a state, this suggests that regardless of the particular value estimate sampled from this distribution, the action selection probabilities were likely to be largely unchanged. Thus, for tractability's sake, we used a single sample from this value estimate distribution to estimate $P(a|s, M, \theta_i)$ for each model class. In order to be fair to the agents whose representations did not include explicit objects, we represented actions here in terms of the available directions (up, down, left, and right). For the agents with object-

based action representations, we simply calculated the first step in the path to each target object and summed together the values of the target objects that involved the same first directional step.

Using this formulation, we fit each agent's parameters θ_i to maximize $\log P(d_i|M, \theta_i)$ for each individual human participant. For these LL parameter fits, we used Bayesian optimization with 20 initial samples and 20 iterations to maximize our estimate of $\log P(d_i|M, \theta_i)$ (Martinez-Cantin, 2014). Since each task example might contain hundreds of discrete steps, we instead estimated the total LL by sub-sampling 100 steps at random from each example when evaluating the sum of the LL for each participant. To ensure that this sub-sampling did not affect our results, we fit some participant parameters both with and without the sub-sampling and observed no qualitative differences in the resulting parameters; as a result, we used sub-sampling to speed up the fitting process.

6. Results and discussion

6.1. Overall performance of human participants and MBP agents

The performance results of the human participants and MBP agents with performance-maximizing parameters are shown in Fig. 6. For clarity, we split the human instances into two categories: training and evaluation. For agents, we show only the average performance on evaluation instances. To evaluate differences in performance across tasks between model classes, we calculated a Bayesian two-sided t test comparing human performance to agent performance on each task. To compute this, we used the standard Jeffreys–Zellner–Siow formulation suggested by Rouder, Speckman, Sun, Morey, and Iverson (2009). Table 2 contains the resulting log Bayes Factors (in base e). Following the guidelines proposed by Jeffreys (1961), Bayes factors greater than 10 (or log Bayes factors greater than 2.3) represent strong evidence that the agent performance was different than human performance on a given task, while factors below 1/10 (or log factors below −2.3) represent strong evidence that agent performance was equivalent to human performance on a task. For convenience, cells in bold represent comparisons for which there was moderate evidence (between 1/3 and 1/10, or log factors between −1.10 and −2.30) that human and agent performance were equivalent.

Overall, these results indicate that once human participants learned the structure of a task, they achieved close to perfect performance on all subsequent instances of that task, even though they had never previously encountered the precise setups of objects in those instances. Given the presence of negatively rewarding objects and complex obstacles, such perfect performance with instance-by-instance task variation suggests that humans did not have to re-learn the roles of objects in each novel instance. Instead, these results suggest that participants came to each new instance of the task with a high degree of certainty about each object's role based on their representation of the task. The performance of the three tabular models in Fig. 6 demonstrates what performance on these tasks would look like if humans were using the baseline, informative-object-features, or relational-

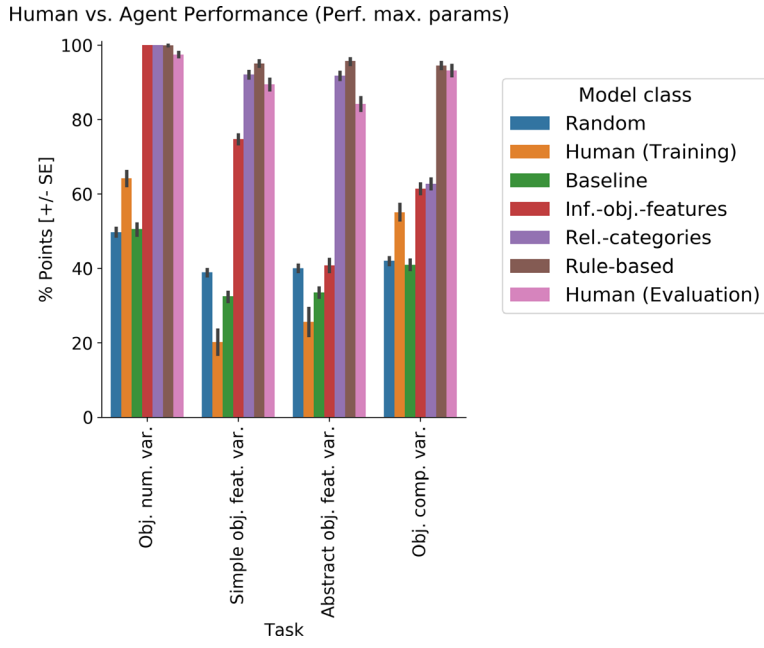

Fig. 6. Performance results by task and model class with performance-maximizing parameters fit. The horizontal axis represents the task, while the vertical axis shows percentage of total possible points earned. Each color corresponds to a model class and error bars reflect ±1 *SE*.

Table 2
Log(Bayes Factor) (LBF) for two-sample test (independent groups). Each cell shows LBF (base e) for performance on one task of one model class versus human. Bold font shows cells where there was moderate evidence of equivalent agent and human performance

Agent Type	Obj. Num. Var.	Simple Obj. Feat. Var.	Abstract Obj. Feat. Var.	Obj. Comp. Var.
Rule-based (Perf.-max)	6.18	3.44	16.13	**1.66**
Rel.-categories (Perf.-max)	7.10	**−1.33**	4.79	83.12
Inf.-obj.-features (Perf.-max)	7.10	26.06	134.25	87.77
Baseline (Perf.-max)	260.72	298.15	208.34	175.76

categories representations and had to update their transition function during these evaluation instances. The baseline, informative-object-features, and relational-categories task representations each failed to reach human levels of performance on one or more tasks. On the other hand, the rule-based representation matched (or exceeded) human levels of success on all four tasks, suggesting that this task representation is capable of accounting for human within-domain flexibility.

6.2. Baseline agent results

The raw, set-of-all-objects-and-features representation used by the baseline agent was highly sensitive to the position and scale of objects, making it too fragile to represent any of our four tasks well. Fig. 7A shows the number of interactions with wall objects and indicates that the baseline agent spent much more of its time bumping into walls than humans or other agents did. These results suggest that the value estimation and planning framework described in Section 2 was not sufficient to account for human behavior by itself.

6.3. Informative-object-features agent results

Filtering out uninformative features allowed the informative-object-features representation to be agnostic to the position and color of objects. However, without the ability to represent relational categories, this agent struggled with the object feature variation and OCVTs. While this representation was able to identify treasures in the ONVT and SOFVT, Fig. 7B shows that in the AOFVT this agent was more likely to try opening doors with decoy keys than with the correct key, suggesting that it was unable to reliably distinguish between keys and decoys when no individual object feature was informative. Furthermore, Fig. 7A shows that fully relational nature of the abstract version of this task caused the informative-object-features representation to regress to the same inability to distinguish between object kinds (including walls) exhibited by the baseline agent. Finally, the inability to represent the outcome of interacting with a conveyor belt independently of that conveyor belt's context also prevented the informative-object-features representation from performing well on the OCVT. This inability can be seen in Fig. 7C, which shows the number of interactions with conveyor belt objects in the OCVT as a function of whether that interaction ultimately led to a treasure, trap, or nothing. On this task, the informative-object-features agent simply avoided conveyor belts entirely, as it could not correctly predict whether they would lead to reward or punishment. The relative ability of the relational-categories representation in the AOFVT and rule-based representation in the AOFVT and the OCVT demonstrate that the planning framework used by these models was capable of succeeding at these tasks, suggesting that the failure of the informative-object-features representation here was a result of the task representation specifically. This suggests that humans are not representing these tasks solely in terms of informative object features or combinations of features.

6.4. Relational-categories agent results

The ability to represent relational categories allowed the relational-categories agent to perform above human level on the object number variation and OFVTs, but was not sufficient to match human performance on the OCVT. Representing object categories in terms of relational propositions allowed this agent to be robust to changes in the position and scale of objects and distinguish keys from decoys in both the simple and abstract OFVT (see Fig. 7B). However, even with the accurate category labels, the representation that this agent used to model this task did not allow it to identify which context-independent combinations of objects worked together to produce positive reward in the OCVT. In this

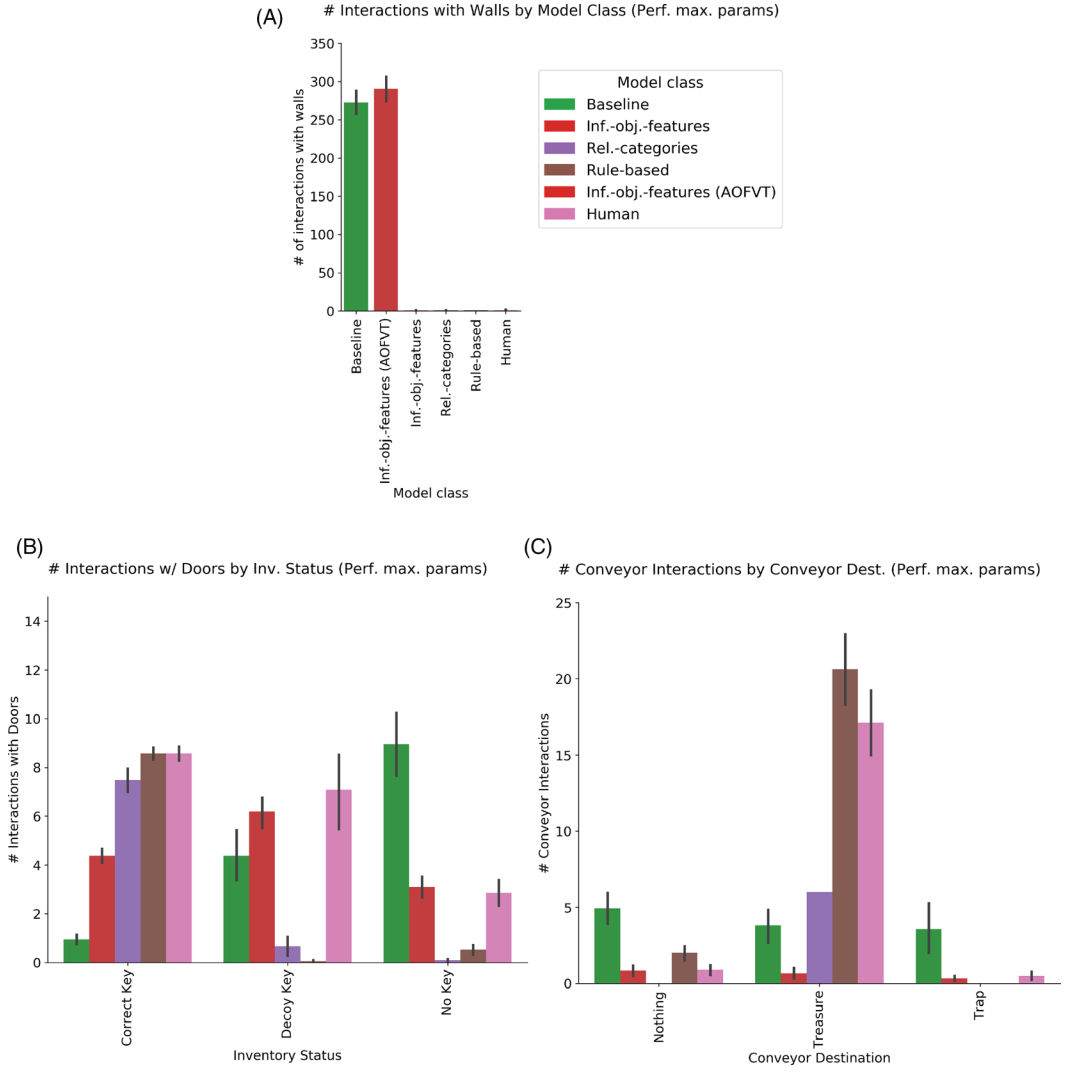

Fig. 7. (A) Number of object interactions that were with walls by model class. The counts are aggregated across all four tasks for each model class, with the exception of the "Inf-obj-features (AOFVT)" model class, which only includes counts from the AOFVT task. (B) Number of door interactions with different inventory statuses in the AOFVT. (C) Number of conveyor belt interactions by outcome in the OCVT. Error bars reflect ± 1 *SE*.

task, the relational-categories representation had just as much difficulty distinguishing helpful conveyor belt combinations from harmful ones as the baseline and informative-object-features models did. Fig. 7C shows that, as a result of this, the relational-categories agent simply learned to avoid interacting with any conveyor belts at all due to being unable to predict the outcomes of these interactions. The overall pattern of results

exhibited by this task representation suggests that humans are also not representing these tasks solely in terms of object categories defined over features and relational propositions.

6.5. Rule-based agent results

In addition to attaining high performance on all four tasks, the rule-based representation also demonstrated qualitatively similar behaviors to humans in each task. In particular, this rule-based representation avoided wall objects (see Fig. 7A), interacted with keys more than decoys (see Fig. 7B), and was capable of properly identifying which arrangements of conveyor belts would lead to reward instead of punishment (see Fig. 7C). However, there were a number of ways in which this rule-based model behaved differently than humans on these tasks. Fig. 7B shows that although the rule-based model interacted with doors while holding the correct key as frequently as human participants did, it rarely interacted with doors while holding a decoy key or no key, whereas human participants did this quite often. Fig. 7C shows that the rule-based agent visited more conveyor belts that lead to treasure than humans did, perhaps because humans simply failed to see some options on the more complicated game screens. Additionally, Table 2 shows that when combined with the planner described in Section 3, the rule-based representation outperformed humans on three out of the four tasks. This suggests that the performance-maximizing parameters that we used to highlight the effect of different representations may have failed to capture some of the planning limitations exhibited by humans.

6.6. Likelihood-maximizing parameter results

To explore the possibility that the performance-maximizing parameters may have failed to capture human planning limitations in more detail, we also fit the planning parameters of our agents to maximize the overall likelihood of the observed data for each individual participant. For a quantitative assessment of each model class's ability to capture human behavior in this task, we also fit each agent's parameters to maximize the LL of the human-selected actions as described in Section 5.3. Table 3 shows the LL estimates of the human data given each model class, separated by task, with these LL-maximizing parameters. Since each model class had the same number of parameters, we can directly compare these LL estimates.

Table 3
Log-likelihood (LL) of selecting the same directional actions as humans for each model class. The first four columns show LL (base e) for each task, while the last column shows total LL across all tasks. Bold font indicates the best-fitting model

Agent Type	Obj. Num. Var.	Simple Obj. Feat. Var.	Abstract Obj. Feat. Var.	Obj. Comp. Var.	Total
Rule-based	−1,028.96	**−5,133.76**	**−7,391.16**	**−3,992.47**	**−17,546.35**
Rel.-categories	**−1,008.50**	−5,359.00	−8,521.56	−5,607.17	−20,496.23
Inf.-obj.-features	−1,011.31	−5,568.18	−13,933.12	−6,545.24	−27,057.85
Baseline	−6,360.35	−11,055.92	−12,283.53	−9,050.28	−38,750.06

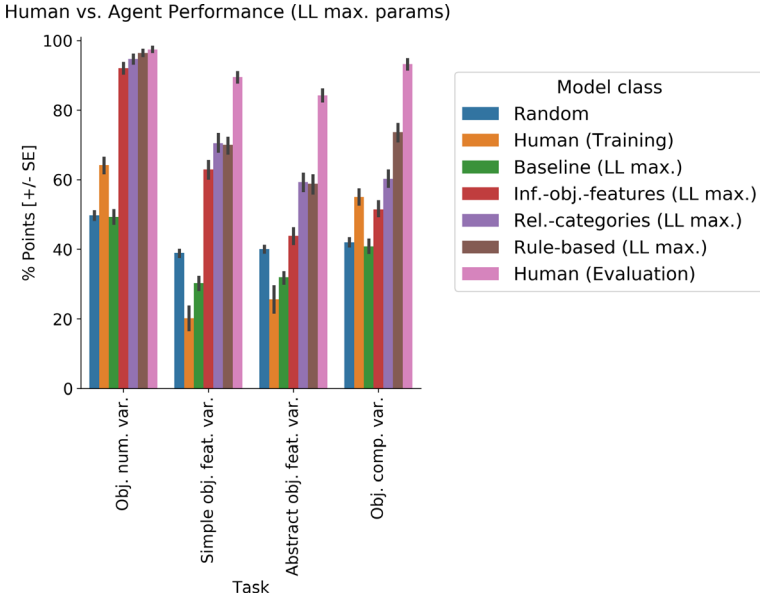

Fig. 8. Performance results by task and model class with parameters fit to maximize the log-likelihood of the human participant data. The horizontal axis shows the task, while the vertical axis shows the percentage of total possible points earned. Each color corresponds to a model class and error bars reflect ± 1 *SE*.

As a form of posterior predictive check, we also simulated how each agent would perform when using these LL-maximizing parameters in place of the performance-maximizing parameters of Fig. 6. In theory, if each agent was predicting human behavior well given these LL-maximizing parameters, the agent should be able to achieve closer to human-like performance when allowed to make its own decisions given those same parameters. Fig. 8 shows the results of these simulations and Fig. 9 shows more detailed information about each agent's interaction with different object kinds. Table 4 shows the log Bayes Factor statistics for Bayesian *t* tests comparing each model class's performance to humans.

Finally, Fig. 10 shows the distribution of these LL-maximizing parameters across participants. For the sake of brevity, we have only provided the parameter distributions for the agent with the rule-based representation, as the distributions for the other agents were qualitatively similar. The horizontal dashed lines on this figure represent the performance-maximizing parameters used to generate Fig. 6. Fitting parameters to maximize LL resulted in overall higher temperature values (τ), resulting in more frequent random actions from the agents. The remaining parameter distributions were bimodal, to varying degrees, with means around the minimum and maximum values for each parameter. There was no significant correlation between any of the parameter values within participants.

Interestingly, the agent performance with the maximum-likelihood parameters suggests that the planning component of our framework may not be well suited to capturing lower

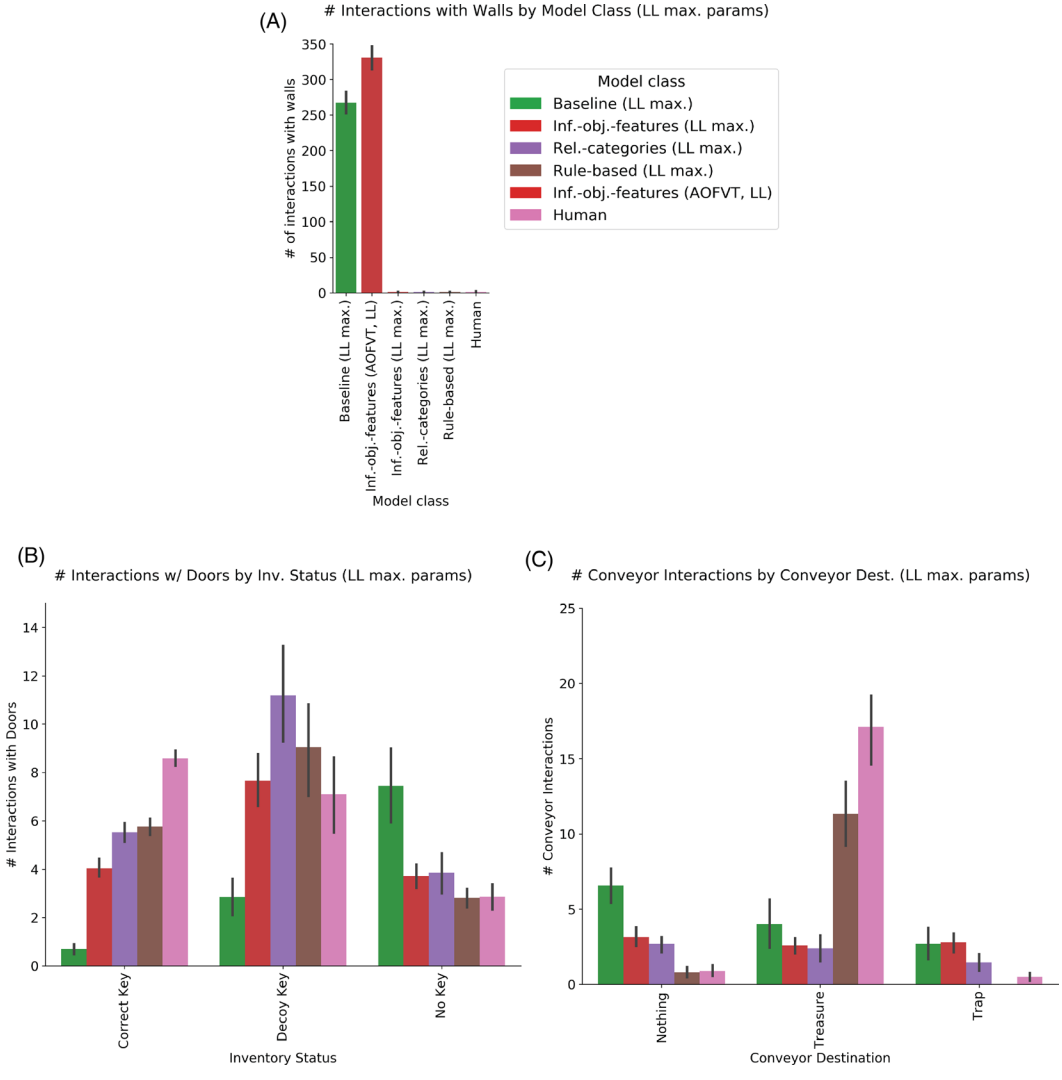

Fig. 9. (A) Number of object interactions that were with walls by model type with LL-maximizing params. The counts are aggregated across all four tasks for each model class, with the exception of the "Inf-obj-features (AOFVT)" model class, which only includes counts from the AOFVT task. (B) Number of door interactions with different inventory statuses in the AOFVT with LL-maximizing params. (C) Number of conveyor belt interactions by outcome in the OCVT with LL-maximizing params. Error bars reflect ± 1 *SE*.

level details of how humans perform this task. While the performance-maximizing parameters caused the agents with rule-based representations to outperform humans on three out of four tasks, Fig. 8 and Table 4 show that the likelihood-maximizing parameters caused the same agents to significantly underperform on most tasks. More specifically, Fig. 9b shows that the likelihood-maximizing parameters caused the rule-based and

Table 4
Log Bayes Factor (LBF) for two-sample test (independent groups) for performance with LL-maximizing parameters. Each cell shows LBF (base e) for performance on one task for one model class versus human performance. Bold font shows cells for which there was moderate evidence for equivalent agent and human performance

Agent Type	Obj. Num. Var.	Simple obj. Feat. Var.	Abstract Obj. Feat. Var.	Obj. Comp. Var.
Rule-based (LL-max	**−1.80**	24.51	30.13	19.70
Rel.-categories (LL-max)	−0.25	19.89	31.52	56.04
Inf.-obj.-features (LL-max)	3.92	40.69	89.35	82.44
Baseline (LL-max)	214.76	220.82	164.94	121.50

relational-categories task representations to interact with doors while holding decoy or no keys at rates that were more similar to humans. However, the same parameters also caused these agents to use keys correctly less than humans did, resulting in the inferior overall performance shown in Fig. 8. In the OCVT, Fig. 9c shows that the likelihood-maximizing parameters caused the rule-based representation to under-visit rewarding conveyor belts while trying to replicate participants' behavior in visiting punishing conveyor belts. In other words, it seems that the likelihood-maximizing parameters caused the agents to over-fit to suboptimal human behaviors, thus leading to overall worse performance on these tasks.

Taken together, these results suggest that the particular parameters used by MCTS and Sarsa (λ) for value estimation may fail to capture some key aspect of human behavior in this task, and thus are difficult to precisely fit to behavior. For example, participants' tendency to interact with doors while holding decoy keys in the OFVT might be a result of cognitive limitations like mistaking the color of one key for something else or forgetting which key they have currently picked up. Once a participant realizes they have made a mistake, they may be able to modulate their attention or increase their focus to improve their performance, which our current planning framework cannot account for. Alternatively, humans simply may not find the extra time required to pick up decoy keys and test them to be as costly as the agents do. Either way, the LL-maximizing parameters seem to account for much of this behavior by increasing the temperature parameter of the agents and thus making the agent's behavior more generally noisy.

In its current form, our framework may also be overly sensitive to the order in which participants interact with objects in these tasks. The framework we have used here attempts to minimize steps taken while maximizing points earned, which leads it to prefer shorter paths between target objects. However, humans may employ high-level target selection strategies like starting with objects at the top of the screen, or moving from the center outwards that might result in longer paths between objects, but similar overall point totals within each task. This could potentially explain the bimodality of the fitted parameter distributions seen in Fig. 10 and the lack of correlation between parameters within participants. If our model is missing elements of human planning like these high-level target selection strategies, then the parameter fitting process may be trying to make

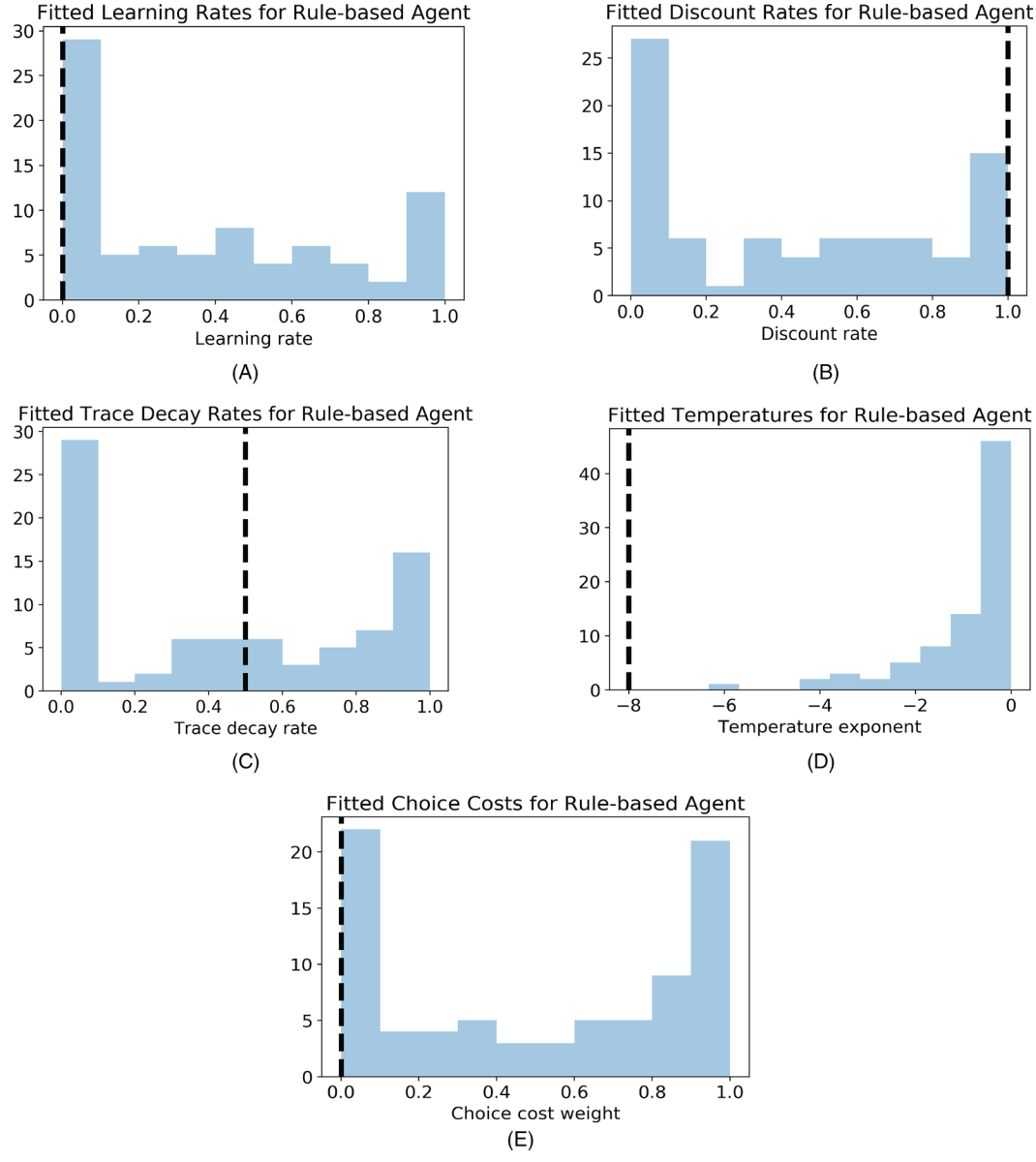

Fig. 10. Distribution of log-likelihood-maximizing parameters across participants, by parameter type. Dashed lines represent performance-maximizing values. (A) The distribution of learning rate; (B) the discount rate; (C) the trace decay rate; (D) the natural log of the temperature for the softmax policy; (E) the cost of making choices from the reward function.

up for these missing elements by making semi-random adjustments to the few parameters it does have. However, while this suggests that we have not provided a complete account of human planning, there is no obvious reason to think that the limitations of this planner

would disproportionately affect any one representation over the others. Thus, given that each representation was provided with the same planner, the differences in cumulative LLs between representations shown in Table 3 still provide evidence for humans using rule-based representations over relational categories or informative-object-features representations.

7. General discussion

The central aim of this work was to explore human generalization across within-domain variation. Drawing on existing work on cross-domain transfer and perceptual/motor invariance, we argued that flexible within-domain transfer in humans stems from the form of representational abstraction used during MBP. Using the computationally tractable domain of video games, we sought to characterize the nature of this abstraction. To do this, we described a general MBP framework capable of playing grid-based video games and proposed four potential task representations that might account for human generalization within this framework. We then evaluated the within-domain flexibility of these four task representations relative to human performance using four novel tasks.

While the recent successes of object-feature-vector- or relational-category-based representations has demonstrated the power of these representational formats (Higgins et al., 2018; Zambaldi et al., 2018) in video game domains, the overall performance of these representations on the tasks described in this work suggests that they are not sufficient in and of themselves to account for the many varieties of within-domain generalization exhibited by humans. Instead, our results suggest that flexible within-domain transfer in humans is better accounted for with representations that combine objects and relational categories with propositional rules. Beyond the general claim that humans reason with propositional rules, we have provided an empirical demonstration of how the various components of object-oriented, relational, and rule-based representations allow for generalization across specific kinds of within-domain variation. This empirical evaluation of the precise nature of the representational abstractions at play in MBP and within-domain transfer is the central contribution of this work.

The claim that human generalization in complex tasks is aided by factored, rule-based representations is convergent with a number of existing lines of research. A wide body of work has shown that young children organize concepts into theory-like representations that allow for causal reasoning (Carey, 2004, 2009; Gopnik et al., 2004; Gopnik & Meltzoff, 1999; Gweon, Tenenbaum, & Schulz, 2010; Schulz, 2012; Wellman & Gelman, 1992, 1998), and that children use a process similar to that of scientific discovery to learn intuitive theories from their environment (Cook, Goodman, & Schulz, 2011; Gweon et al., 2010; Schulz, Gopnik, & Glymour, 2007; Stahl & Feigenson, 2015; Tsividis, Gershman, Tenenbaum, & Schulz, 2013). Studies of human learning in game environments have found that humans also use richly structured theories when learning in complex sequential decision tasks (Dubey et al., 2018; Tsividis et al., 2017), and it has been argued that these kinds of core cognitive elements are necessary for replicating human-

like intelligence in AI systems (Lake, Lee, Glass, & Tenenbaum, 2014; Lake et al., 2017). While this existing work has identified theories as central to cognition, a precise formal definition of what constitutes a theory or how it can be used to drive problem-solving has remained elusive. One possibility is that the broader notion of "theory" referenced in this existing work can be mapped onto something like the factored, rule-based transition functions within an MBP framework similar to that presented here.

There are a number of key limitations to the current work. First, we have only explored a small set of the many kinds of within-domain variation that humans can flexibly handle. While the three specific forms of variation that we have selected here are present in a wide array of natural tasks that humans face every day, they clearly do not cover the entire space of cognitive invariances demonstrated by humans. Furthermore, the current work does not explore any variations that highlight the limitations of human task representations. A robust understanding of human problem-solving requires identifying the dimensions of variation that humans find challenging as well as those they find intuitive. For this reason, we intentionally designed the grid-based game system used in this work to be flexible enough to design tasks that vary in more challenging ways. It is an interesting open question for future work what these more challenging tasks might look like.

A second limitation of the current work is that we have only explored one potential MBP framework here. Even within the space of RL approaches MBP, there are a large number of different implementations that might better account for individual variation in humans (Kaiser et al., 2019; Schrittwieser et al., 2019; Zhang et al., 2019). It would be particularly interesting to explore agents equipped with more robust elements of HRL in future work. As discussed in the model descriptions, we did provide our agents with temporal abstractions in the form of object- and relational-category-based subgoals, but we did not fully explore the range of redundant action patterns that a more robust HRL agent might extract in learning to solve these tasks. For example, almost every instance of the OFVT involved picking up a key and using it to open a door, which could feasibly be compressed into a more abstract "open door" action. It is possible that part of the reason that human behavior is not perfectly predicted by the planning algorithms presented here is that humans are in part reusing options or high-level strategies from previous tasks (e.g., "start with the objects at the top and work downwards"), while the algorithms we used here simply try to minimize steps while maximizing points. Recent work on deep HRL and compositional policy reuse has demonstrated impressive successes in regard to learning these kinds of more abstract action representations (Kulkarni, Narasimhan, Saeedi, & Tenenbaum, 2016; Wingate, Diuk, O'Donnell, Tenenbaum, & Gershman, 2013). It is an interesting question for future work whether cutting-edge HRL or policy-reuse approaches would be able to extract strategies that would better account for the low-level details of how humans solve our games.

Outside of RL approaches to MBP, the AI community has focused recently on the strength of various neural network approaches to solving complex tasks. A large amount of attention has been devoted to research on neural network implementations of problems like representation learning (Bengio et al., 2013), sequential decision tasks (Mnih et al.,

2015; Silver et al., 2016, 2017), and multitask learning (Wang et al., 2018). One of the challenges of using these kinds of approaches for studying human learning in particular is that the internal dynamics of such systems are often difficult to interpret. For example, if we trained a neural network to produce human-like behavior across all four of the tasks in this paper, it would still be difficult to say exactly what kind of representation was being used by the network in order to produce this behavior. Work on neural turing machines has shown that given the proper architecture and training, neural networks can replicate Turing machines, suggesting that they can, in theory, replicate any of the representations we have described here (Graves, Wayne, & Danihelka, 2014). However, just because it is theoretically possible to instantiate such dynamics in a neural network does not mean it can be done within a feasible set of computational and temporal constraints. Thus what kind of training regimen and neural network architecture could actually learn to produce human-like robustness on our tasks remains an interesting open question. It is worth noting that we intentionally constructed each of our tasks to be able to produce a near-infinite number of unique training instances in order to provide the large training sets often required by these kinds of approaches.

In terms of symbolic approaches, there are many general problem solvers that also attempt to take abstract task representations and identify potential solutions, some of which operate on rule-based representations that look similar to those used here (for an in-depth review of these approaches, see Kotseruba & Tsotsos, 2016). In order to make our results as algorithm-agnostic as possible, the tasks we have included here were designed to have a small set of desirable solutions of relatively short length. Thus, given the appropriate task representation and enough time to plan, we would expect most planning systems to return a plan from this small set. However, as pointed out in Section 6, the parameters of the planner that we have used here do not seem to account well for individual variation between participants, and it would be interesting to explore to what extent other planners might be able to better account for individual variation. Furthermore, extensive planning is often computationally expensive, and many real-world situations require rapid responses that would preclude the ability to plan with many simulations. Therefore, it would also be interesting to explore the differences between planners when the number and breadth of simulations is limited.

Finally, the current work is largely agnostic to the question of how humans learn the kinds of task representations that we have examined here. Instead, the current work focuses on illuminating the output of these learning processes, which is an important first step toward formalizing a full computational account of the learning process itself. In future work, we hope to address the question of what kind of inductive biases might allow humans to learn these abstract task representations as quickly as they do.

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),

Google, and the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. We are grateful to Matt Botvinick and Josh Tenenbaum for many insightful discussions.

Notes

1. This representation is Markovian in the sense that it assumes that the probability of transitioning to a given state is only dependent on the current state and the action taken, not on any previously visited states or actions. While many real-world environments contain dependencies between temporally distant states that seemingly violate this Markovian assumption, these dependencies can often be accounted for within the framework of an MDP by introducing latent variables describing these dependencies into the state representation. Whether inferring such latent variables is computationally tractable in real-world environments is an interesting open question that is outside the scope of the current work.
2. We also evaluated letting humans navigate step-by-step by using the up, down, left, and right keys, but saw no qualitative difference in performance.

References

Anderson, J., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological Review*, *126*, 727–760.

Bassok, M., & Holyoak, K. (1989). Interdomain transfer between isomorphic topics in algebra and physics. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*, 153–166.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Carey, S. (2004). Bootstrapping and the origin of concepts. *Daedalus*, *133*(1), 59–68.

Carey, S. (2009). *The origin of concepts*. New York: Oxford University Press.

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

Cook, C., Goodman, N., & Schulz, L. (2011). Where science starts: Spontaneous experiments in preschoolers' exploratory play. *Cognition*, *120*(3), 341–349.

Diuk, C., Cohen, A., & Littman, M. (2008). An object-oriented representation for efficient reinforcement learning. In A. McCallum & S. Roweis (Eds.), *Proceedings of the 25th international conference on machine learning (ICML)* (pp. 240–247). New York: Association for Computing Machinery.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., & Efros, A. A. (2018). Investigating human priors for playing video games. *Proceedings of the 35th International Conference on Machine Learning, PMLR*, *80*, 1349–1357.

Duncker, K. (1945). On problem-solving. *Psychological Monographs*, *58*, i–113.

Garnelo, M., & Shanahan, M. (2019). Reconciling deep learning with symbolic artificial intelligence: Representing objects and relations. *Current Opinion in Behavioral Sciences*, *29*, 17–23.

Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, *15*, 1–38.

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*(1), 3–32.

Gopnik, A., & Meltzoff, A. (1999). *Words, thoughts, and theories*. Cambridge, MA: MIT Press.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. https://arxiv.org/abs/1410.5401

Gweon, H., Tenenbaum, J., & Schulz, L. (2010). Infants consider both the sample and the sampling process in inductive generalization. *Proceedings of the National Academy of Sciences of the United States of America*, *107*(20), 9066–9071.

Halford, G., Wilson, W., & Phillips, S. (2010). Relational knowledge: The foundation of higher cognition. *Trends in Cognitive Science*, *14*(11), 497–505.

Hengst, B. (2012). Hierarchical approaches. In M. Weiring & M. V. Otterlo (Eds.), *Reinforcement learning: State of the art* (pp. 293–323). Berlin, Germany: Springer-Verlag.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., & Lerchner, A. (2018). Darla: Improving zero-shot transfer in reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 1480–1490.

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford, UK: Oxford University Press.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., & Michalewski, H. (2019). Model-based reinforcement learning for Atari. https://arxiv.org/abs/1903.00374

Kaplan, C. A., & Simon, H. A. (1990). In search of insight. *Cognitive Psychology*, *22*, 374–419.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *European conference on machine learning* (pp. 282–293). Berlin, Germany: Springer-Verlag.

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

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from Tower of Hanoi. *Cognitive Psychology*, *17*, 248–294.

Kotseruba, I., & Tsotsos, J. K. (2016). A review of 40 years of cognitive architecture research: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*, 17–94.

Kulkarni, T., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. D. Lee, U. von Luxburg, R. Garnett, M. Sugiyama, & I. Guyon (Eds.), *30th conference on neural information processing systems* (pp. 3682–3690). Red Hook, NY: Curran Associates.

Lake, B., Lee, C., Glass, J., & Tenenbaum, J. (2014). One-shot learning of generative speech concepts. In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of the 36th annual conference of the Cognitive Science Society* (pp. 803–808). Red Hook, NY: Curran Associates, Inc..

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

Lang, T., Toussaint, M., & Kersting, K. (2012). Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research*, *13*, 3725–3768.

Luce, R. (1959). *Individual choice behavior*. New York, NY: John Wiley and Sons Inc.

Luchins, A. (1942). Mechanization in problem solving: The effect of Einstellung. *Psychological Monographs*, *54*, i–95.

Martinez-Cantin, R. (2014). Bayesopt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, *15*, 3735–3739.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Mohan, S., & Laird, J. E. (2011). An object-oriented approach to reinforcement learning in an action game. In V. Bulitko & M. Reidl (Eds.), *Proceedings of the seventh AAAI conference on artificial intelligence and interactive digital entertainment* (pp. 164–169). Menlo Park, CA: The AAAI Press.

Piantadosi, S. (2011). Learning and the language of thought (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Ponsen, M., Taylor, M., & Tuyls, K. (2010). Abstraction and generalization in reinforcement learning: A summary and framework. In M. Taylor & K. Tuyls (Eds.), *Adaptive learning agents* (pp. 1–32). Berlin: Springer-Verlag.

Raffin, A., Hill, A., Traoré, R., Lesort, T., Díaz-Rodríguez, N., & Filliat, D. (2019). Decoupling feature extraction from policy learning: Assessing benefits of state representation learning in goal based robotics. In *The proceedings of the workshop on structure & priors in reinforcement learning at ICLR 2019*. https://arxiv.org/pdf/1901.08651.pdf

Rouder, J., Speckman, P., Sun, D., Morey, R., & Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review*, *16*(2), 225–237.

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

Santoro, A., Raposo, D., Barrett, D., Malinowski, M., Pascanu, R., & Battaglia, P. (2017). A simple neural network module for relational reasoning. In 31st conference on neural information processing systems, Long Beach, CA, USA.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2019). Mastering atari, go, chess and shogi by planning with a learned model. https://arxiv.org/abs/1911.08265

Schulz, L. (2012). The origins of inquiry: Inductive inference and exploration in early childhood. *Trends in Cognitive Sciences*, *16*(7), 382–389.

Schulz, L., Gopnik, A., & Glymour, C. (2007). Preschool children learn about causal structure from conditional interventions. *Developmental Science*, *10*, 322–332.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*, 354–359.

Spelke, E. (2003). What makes us smart? Core knowledge and natural language. In D. Gentner & S. Goldin-Meadow (Eds.), *Language in mind: Advances in the investigation of language and thought* (pp. 277–311). Cambridge, MA: MIT Press.

Spelke, E., & Kinzler, K. (2007). Core knowledge. *Developmental Science*, *10* (1), 89–96.

Stahl, A., & Feigenson, L. (2015). Observing the unexpected enhances infants' learning and exploration. *Science*, *348*(6230), 91–94.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tomov, M., Yagati, S., Kumar, A., Yang, W., & Gershman, S. (2020). Discovery of hierarchical representations for efficient planning. *PLOS Computational Biology*, *16*, e1007594.

Tsividis, P. (2019). Theory-based learning in humans and machines (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Tsividis, P., Gershman, S., Tenenbaum, J., & Schulz, L. (2013). Information selection in noisy environments with large action spaces. In P. Bello, M. Guarini, M. McShane, & B. Scassellati. *Proceedings of the 36th annual conference of the Cognitive Science Society* (pp. 1622–1627). Red Hook, NY: Curran Associates.

Tsividis, P., Pouncy, T., Xu, J., Tenenbaum, J., & Gershman, S. (2017). Human learning in Atari. In G. Roig & X. Boix (Eds.), *AAAI spring symposium on science of intelligence: Computational principles of natural and artificial intelligence* (pp. 643–646). Palo Alto, CA: The AAAI Press.

van Opheusden, B., & Ma, W. J. (2019). Tasks for aligning human and machine planning. *Current Opinion in Behavioral Sciences*, *29*, 127–133.

Vodopivec, T., Samothrakis, S., & Ster, B. (2017). On Monte Carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, *60*, 881–936.

Wang, J., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J., Hassabis, D., & Botvinick, M. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, *21*, 860–868.

Wellman, H., & Gelman, S. (1992). Cognitive development: Foundational theories of core domains. *Annual Review of Psychology*, *43*, 337–375.

Wellman, H., & Gelman, S. (1998). Knowledge acquisition in foundational domains. In D. Kuhn & R. S. Siegler (Eds.), *Handbook of child psychology: Vol. 2. Cognition, perception, and language development* (5th ed., pp. 523–573). New York, NY: Wiley.

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

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., & Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., & Battaglia, P. (2018). Relational deep reinforcement learning. https://arxiv.org/abs/1806.01830

Zettlemoyer, L., Pasula, H., & Kaelbling, L. (2005). Learning planning rules in noisy stochastic worlds. In M. Veloso & S. Kambhampati (Eds.), *Proceedings of the national conference on artificial intelligence (AAAI)*. Palo Alto, CA: The AAAI Press.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., & Levine, S. (2019). Solar: Deep structured representations for model-based reinforcement learning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning*. New York, NY: Association for Computing Machinery.

Zhong, G., Wang, L., Ling, X., & Dong, J. (2016). An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, *2*(4), 265–278.

Zhu, G., Huang, Z., & Zhang, C. (2018). Object-oriented dynamics predictor. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, & N. Cesa-Bianchi (Eds.), *32nd conference on neural information processing systems*. Red Hook, NY: Curran Associates.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article:

Appendix A. Task representation implementation details.

Appendix B. Task generation.

Appendix C. Model specification and simulation.