Abductive task abstractions in physical problem-solving without object-level prior

Abstract

Humans solve problems with ease for two primary reasons. For one, we can infer problem context by leveraging object-level prior (henceforth *prior* for brevity), such as semantics, affordance, and physics. For another, we can simplify the problem context to a task-oriented abstract representation (henceforth *task abstraction* for brevity), such as only considering the routine graph instead of the entire city map when traveling by subway. Unfortunately, priors are latent or inaccessible in many real-world problems, thus difficult to separate them from task abstraction. Here we introduce an experimental procedure to probe how people generate task abstraction *without* priors. We eliminate priors by building a physical environment of magnets, which mimics three major families of realistic physical problems, *i.e.*, tool use, discovery, and crafting, with variants on goals. We devise a hierarchical generative model that accounts for human behavior of task abstraction from the first principles. We design experiments that require participants to solve different problems instructed with a goal (Exp1) or goal and constraint (Exp2) to testify model hypothesis. Model fitting on data of Exp1 significantly shows that task abstractions converge on the same family of goals regardless of variants, yet the policies for solving are diverse. Thus, we further show that the diversity of policies may come from personal preference, independent of the generation of task abstraction, echoing the hierarchy of our model. In Exp2, variants of constraint generally do not transform task abstraction, defending the generality of our findings. With the model hypothesis checked and the comparison with alternate accounts, we suggest that with the absence of priors, people may construct task abstraction from common sense on tasks in a top-down manner. Our findings may lead to general discussions on "knowing how to solve an unknown problem before starting to solve it".

Keywords: Physical problem solving, task abstraction

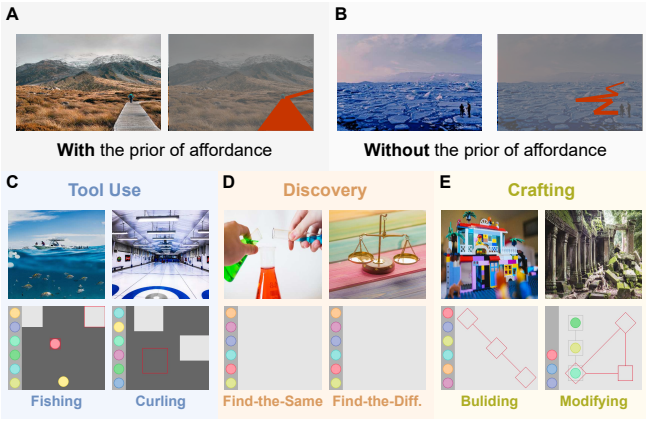

Figure 1: **Illustration of the general problem.** (A) With the object-level prior to affordance, people easily construct a task abstraction on the bridge when navigating to the mountains. (B) Consider navigating to the arctic mountains across drift ice. Since the movement of drift ice is radical and unpredictable, people do not know the affordance in advance. They managed to succeed in navigation, though, by maintaining a task abstraction of "a bridge in logic". They jump by the blocks of drift ice and keep the overall direction toward the mountains. (C-E) Families of problems without object-level prior yet reflect realistic problems in the physical world. Each family consists of 2 variants on goal.

Introduction

People master all kinds of problems in the physical world, such as using tools to extend capability, discovering latent properties of things, and building structures from sketches. Besides the general ability of learning (Bransford et al., 2000), human prior is the most unique and significant factor that facilitates problem solving (Dubey et al., 2018), thus providing people with a sense of knowing how to solve *before* start to solve it (Chu & Schulz, 2020; Pelz et al., 2022). First, people understand problem context from common sense, such as semantics (Dubey et al., 2018), affordance (Gibson, 1979; Zhu et al., 2015), forces and gravity (Battaglia et al., 2013; Ullman et al., 2017; Kubricht et al., 2017; Allen et al., 2020), and social interactions (Jara-Ettinger et al., 2016; Ho, Saxe, & Cushman, 2022). On this basis, people construct problem abstractions by selecting useful information according to current goal (Rigotti et al., 2013), to plan efficiently with limited cognitive resources (Ho, Abel, et al., 2022; Callaway et al., 2022; De Martino & Cortese, 2022). What has been taken for granted is that such information can always be integrated and processed **with ease**. Thus, the major approach to making task construals is filtering out the ones of interest. However, for problems in the physical world, information integration from problem context is **nontrivial**, especially when the properties of objects and environment are latent, *e.g.*, no spatial prior of gravity or no semantic prior of tools. In such conditions, do people learn to solve the problems *tabula rasa*, or still construct problem abstractions based on some other priors?

Our pilot study illustrates such phenomena, where people exploit yet another kind of common sense rather than learning from scratch. Consider a world of magnets on a rough surface. The force of magnetism cannot be predicted given only spatial information of the objects, unlike the forces of supporting and gravity in the normal physical world. Accordingly, without knowing the polarity of the magnets, one cannot get used to the object-centric prior in advance, yet can only learn the properties of the magnets by in-situ observations. Surprisingly, given only task specifications, people *know* how to use the magnets to solve the problem without much exploration in the environment. If the task requires tool use, people pay attention to exploit the interactions between an object, treated as the *tool*, and another object, viewed as the *target*; when asked to search for an object with different polarity to others, people treat objects as experimental groups for observation; once instructed to build a structure, people try to eliminate

the destroying magnetism inside the structure. Though trial-and-error is still required to succeed, it seems that people are guided by those higher-level strategies.

Such strategies may come form people's common sense about task specifications in daily life (see Fig. 1). For example, when instructed with reaching a target object in inaccessible area, people come up with the idea of using tools to extend the capacity of reaching, same may even recall specific events like *fishing*. Thus, instead of **reducing** representation from a collection of information, people may **generate** abstraction by the information transferred from prior experience. Specifically, we suggest that people infer from the abstract task specification in a top-down manner. In contrast, construction representations with object-level prior mostly come in a bottom-up fashion. Thus, we call this kind of abstraction as **task-oriented abductive abstraction**.

In this work, we sought to study how people generate problem abstraction in the absence of a specific problem context, especially the object-centric priors of semantics, affordance, and physics. To isolate task specification from these confounding factors, following the insight in the pilot study, we propose the **ProbSol Worlds (ProbSol)**[1] environment. ProbSol is a physically-grounded 2-D world with magnets on a rough surface. To note, magnetism is *interesting*, for that people are able to reason the physical properties, such as relative polarity and relative mass, through the observation of relative velocity (Ullman et al., 2018)—but people can never know the exact properties of objects only if informed in advance. Such property helps remove the existence of object-level prior. Specifically, ProbSol eliminates confounding factors such as semantic prior in Montezuma's Revenge (Mnih et al., 2015), *e.g.*, fires and monsters that may be harmful, or doors and ladders that may lead to more space for exploration—such prior knowledge specifies problem context by making the properties of objects visible to the solvers; or spatial prior in physical reasoning tasks like Phyre (Bakhtin et al., 2019), *e.g.*, the effect of gravity and elasticity determined by relative positions between objects—such prior knowledge specifies problem context by simulating the dynamics via intuitive physics (Allen et al., 2020). In contrast, ProbSol provides a sandbox-game-like world environment that supports the arbitrary assignment of both semantic and spatial information—different tasks are generally controlled by the same dynamics (magnetism) but become diverse for manipulating task specification.

We decompose task specification into two lines: goals and constraints, following the problem solving literature (Altman, 1999). Normally, goals span a space of *meaningful strategies* while constraints bound the space by *legal strategies*. First goals and then constraints shape a limited space of plausible strategies. On this basis, we specify ProbSol with three families of goals, *i.e.*, *tool use*, *discovery*, and *crafting*; and two families of constraints, *i.e.*, *chances of trial* and *steps of action per trial*. Under each family, there are several instantiated variants to maintain generality. We introduce a hierarchical generative account to explain the mechanism of task-oriented abductive abstraction. We formally describe and discriminate the abstractions for different families of goals from the first principles. Given task specification of goals, human behavior shows that such abstractions are formed from the very beginning—the abstractions are **converged** according to the family of goals, though given different variants of the same goal and generated by different individuals. Under similar problem abstractions, there are multiple plausible policies to reach the goal. People trade-off over policies based on personal preference of value, such as aggressive or conservative, **diversify** the policies. Cross the families and variants of constraints, the same task abstractions are maintained according to goal specification. Our account fits human data well, suggesting that people generate task abstraction in a top-down manner based on a prior understanding of task specification.

[1] Visit `probsol.yzhu.io` for the web-based user interface.

Methods

The ProbSol Worlds environment

We implemented a web-based version ProbSol for behavioral studies. The physical world of magnets is built upon `Chipmunk-js`[2] and is rendered by `GameJS`[3]. A python equivalent of the web-based version is also implemented for model simulations, implemented by `PyMunk`[4] and `PyGame`[5]. In the web-based version, besides the window of the physical environment, there is a store bar on the left to store objects. Input from the mouse is recorded as a step of action. A re-initialization in the same task is recorded as a chance of trial.

Goal specification We create two variants of specific goals under every one of the three families of goals. Instead of simply changing environmental information (*e.g.*, spatial) and object-centric information (*e.g.*, physical properties of objects), each one of the variants seems very different from the other, yet the two share similar abstractions—this design helps separate abstraction level and policy level.

Tool Use, such as fetching targets from inaccessible areas to manipulable areas with the help of other objects, reflects the capability of creative tool use and planning in both humans and intelligent animals (Osiurak & Badets, 2016; Bird & Emery, 2009). The two specified goals are analogous to **Fishing** and **Curling**. Discovery, finding the only pair of objects that share the same property from the given set of objects, reflects the capability of causal inference in both humans and intelligent animals (Schulz, 2012; Blaisdell et al., 2006). The two specified goals are analogous to **Find-the-Difference** and **Find-the-Same**. Crafting, such as arranging objects to resemble a given shape or transform a shape into another, reflects the capability of concept sketching and synthesis in humans (Lake et al., 2015; Fan et al., 2020). The

[2] `https://github.com/josephg/Chipmunk-js` [3] `https://gamejs.org/` [4] `https://www.pymunk.org/` [5] `https://www.pygame.org/`

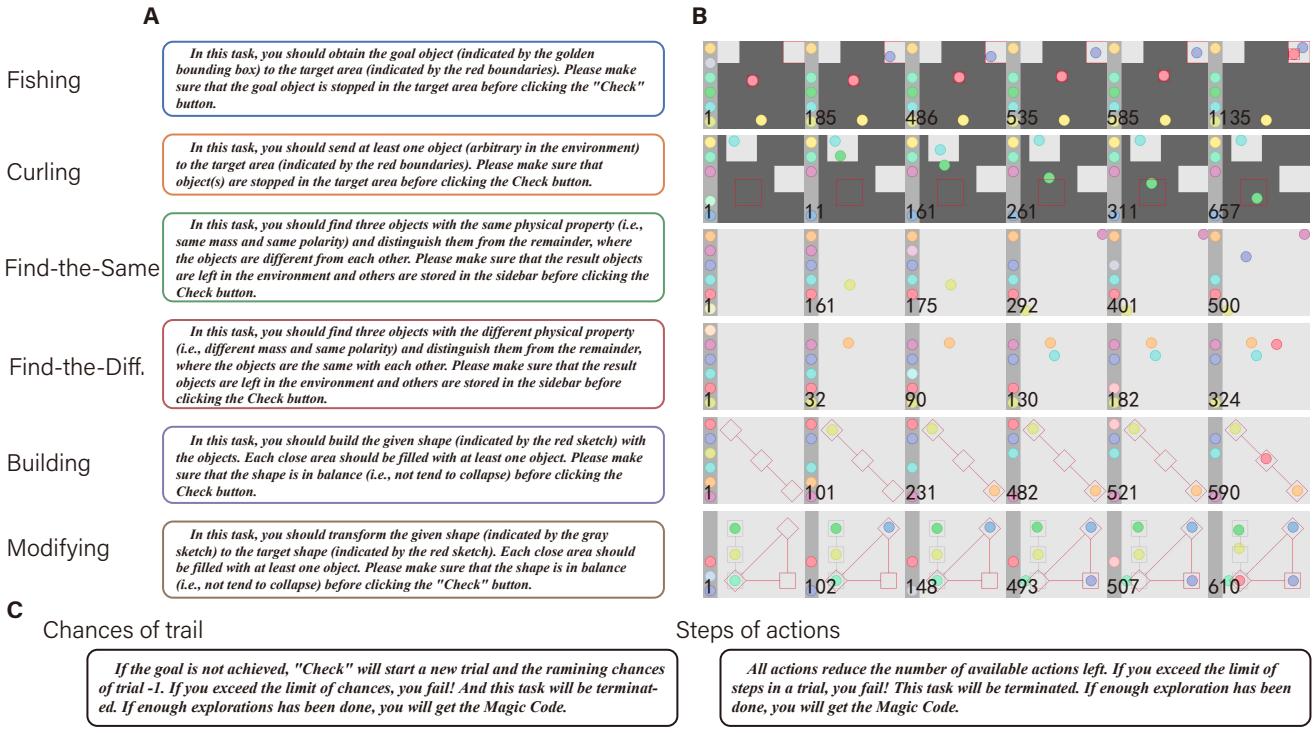

Figure 2: **Overview of the ProbSol Worlds environment.** The simulated environment mimics physically-grounded problems in daily life. Circles denote objects, and areas with dark backgrounds indicate the inaccessible area. Six rows aligned at the left show overviews of the 6 goal specifications: Fishing, Curling, Find-the-Same, Find-the-Difference, Building, and Modifying. (A) The language instructions given to participants as goal specifications. (B) Visual demonstration of the 6 goal specifications with representative solutions. These sequences are manually produced. The number in the lower left corner represents the number of image frames (60 frames in total for 1s). (C) The language instructions given to participants as constraint specifications.

Table 1: **Constraint specifications assigned to goal specifications.** The constraint in the domain (horizontal) is assigned according to the distribution on that indicator in pilot study.

		Chances of trail			Steps of action		
		Easy	Normal	Hard	Easy	Normal	Hard
ToolUse	Fishing	8	3	1	30	12	6
	Curling	6	3	1	70	45	6
Discovery	Fishing	15	3	1	60	40	6
	Curling	15	5	1	60	20	6
Crafting	Fishing	6	3	1	45	30	15
	Curling	6	3	1	30	15	8

two specified goals are analogous to **Building** and **Modifying**. See Fig. 2B for a visual demonstration.

Constraint specification We create three variants of specific constraints under each one of the two families of constraints. All variants are categorized into three types, **Normal**, **Easy**, and **Hard**. The concrete limits on chances of trial and steps of action per trial are preregistered according to the pilot study. All tasks in the pilot study are only specified with goals. The times of trial and steps of action per trial from the trajectories of all participants on all goals are fitted into a Gaussian distribution respectively. For each goal, **Normal** is selected by the mean of the Gaussian model; **Easy** is selected by $+3\sigma$ of the Gaussian; and **Hard** is selected by -3σ of the Gaussian, see Tab. 1 for details. Thus, the tasks with constraint specification would be neither intractably hard to solve nor totally degraded to unconstrained.

Shared procedure In all experiments, including the pilot study, participants are provided with a mandatory video tutorial on how to interact with the ProbSol interface. In the tutorial, all the atom actions, such as "Left-click the mouse on a placed object to grab it", are covered with detail. Following is a free-play period—participants are asked to freely explore a ProbSol environment configured with objects. This period lasts for at least half a minute. Finally, the participants are required to pass a comprehensive quiz with all questions answered correctly. Those who failed to pass the quiz two times are excluded. This pipeline makes sure that the participants filtered by the exclusion criteria are familiar with the basic interactions to eliminate the factor of interface usage from performance. In the shared part of this pipeline, participants are never informed of any task specifications in the testing period. All participants are recruited by Prolific and are from the UK and the USA. They are paid an hourly wage of £8.01 and a £0.5-2 bonus based on their success rate.

Goal-specified experiments Besides the shared tutorial, participants are educated with a very simple example of goal specification, "Putting one object into the close area". By reaching this goal, the tutorial shows the concepts of "What happens when the goal is reached?" and "How to restart a trial?". Questions about such rules are also included in the comprehensive quiz. Every participant is required to solve three tasks. At each task, the participant is given a short instruction in natural language as task specification (see Fig. 2A for details) right before jumping to the page of environment. The goals of the tasks are assigned with the three families of goals respectively, and every goal is specified with one of the variants. The sequence of task specifications is randomized to reduce the effect of fatigue accumulation. 87 complete submissions have been received. Data from $n = 75$ participants are taken into analysis after exclusions.

Goal- and Constraint-specified experiments Besides the shared tutorial, participants are educated with a very simple example of goal-and-constraint specification, "Putting one object into the close area within five steps per trial". By reaching this goal with satisfying this constraint, the tutorial shows the concepts of "What happens when the goal is reached?", "What happens when the constraint is violated?", and "How to restart a trial?". Questions about such rules are also included in the comprehensive quiz. Every participant is required to solve three problems. Besides goal specification, the participant is given additional constraint specification (see Fig. 2C for details). The goals of the tasks are assigned with the three families of goals respectively, and every goal is specified with one of the variants. The three tasks are all specified with only one variant in one of the two family constraints. The sequence of task specifications is randomized to reduce the effect of fatigue accumulation. 188 complete submissions have been received. Data from $n = 166$ participants are taken into analysis after exclusions.

Modeling abductive task abstraction

We propose a hierarchical Bayesian account for modeling abductive task abstraction from the first principles. Let t be task specification and θ be task abstraction. As task abstraction is generated by the common sense understanding of the task specification that comes from prior experience, we view t as a latent variable that generates θ, thus:

$$p(\theta,t) = p(t)p(\theta|t). \tag{1}$$

Denote π as the policy of solving problems and λ as personal preference. On the basis of our hypothesis, a policy is subject to task abstraction and is also generated by personal preference. Hence, we have $p(\pi,\theta,\lambda) = p(\theta,\lambda)p(\pi|\theta,\lambda)$. As we assume λ and θ are independent, we have $p(\pi,\theta,\lambda) = p(\pi|\lambda,\theta)p(\lambda)p(\theta)$. By applying the chain rule, our hierarchical generative model can be formulated as:

$$p(\pi,\lambda,\theta,t) = p(\pi|\lambda,\theta)p(\lambda)p(\theta|t)p(t). \tag{2}$$

Our methodology for result analysis would fit human data with the model through the posterior $p(\lambda,\theta,t|\pi)$. We probe the components of interest separately and discuss the role they are playing in the full model. Following are the implementation details of the model.

Implementation of task abstraction In physical problem solving, there are two major schools of representation: object-centric representation (Allen et al., 2020) and relation-centric representation (Hafri & Firestone, 2021). Graph, combining the merits of both, is believed to be a more general and expressive representation (Tenenbaum et al., 2011)—objects can be represented as nodes, pairwise relations on the cross space of all objects can be represented as edges, and temporal change can be described by an extended dimension of sequential frames. Further, cliques and supernodes enable the representation of different abstraction levels, which is specially required for modeling physical environments. Thus, we integrate some language features in the cutting-edge graph-query programming languages, Cypher (Francis et al., 2018) and GraphQL (Hartig & Pérez, 2018), into the conventional second-order language (Emde et al., 1983). The introduced features enable us to *pay attention to* objects of arbitrary number as well as interacting with some of them. Specifically, we define an operation called `abs` that:

$$\begin{aligned} &\texttt{abs}(G' = \langle V', E' \rangle, \texttt{op}) := \forall v, e \in V', E' \; \texttt{op}(v) \,|\, \texttt{op}(e) \\ &\quad \texttt{op} := \texttt{P(X)} \,|\, \texttt{Q(X,Y)} \,|\, \texttt{op(X)}, \texttt{Q(Y,X)}, \end{aligned} \tag{3}$$

where G' is a subset of the fully connected graph G over all objects and relations in the world, `op` is a highly flexible abstract operator that can be applied to both objects or relations through pattern matching, `P`, `Q` are second-order predicates that can be grounded to specific operations—the operations can either be observation or intervention. `Q` can be both undirected (*i.e.*, `X`, `Y` are in a set of arguments) or directional (*i.e.*, `X` and `Y` are in a partially-ordered set of arguments). This abstract representation spans a very large plausible space of task abstraction $p(\theta|t)$.

Generating and grounding task abstraction As We aim to fit $p(\theta|t)p(t)$ with human data, as $p(t)$ denotes the people's common sense in understanding task specification, we first define a preregistered model to describe task specifications on three families of goals. Task abstraction of Tool Use is `op=act(X),dir(X,Y)`, where `act` is a unary intervention, and `dir` is a dyadic and directional observation on relative position. Task abstraction of Discovery is `op=op(X),rel(X,Y),act(Y)`, where `rel` is a dyadic observation on relative velocity, and the tail-recursive structure indicates that maybe more than two objects are involved in the task construal. Task abstraction of Crafting is `op=mov([X]),cha([X]),exc([X],[Y])`, where `mov` is an observation on absolute pace for an array of objects `[X]`, `cha` is an intervention of exchanging two objects in `[X]`, and `exc` is an intervention of exchanging two objects between `[X]` and `[Y]`.

Since trajectories generated by participants are highly noisy, we ground the task abstractions to a 5-dimensional feature space. Thus our model is able to draw statistical features

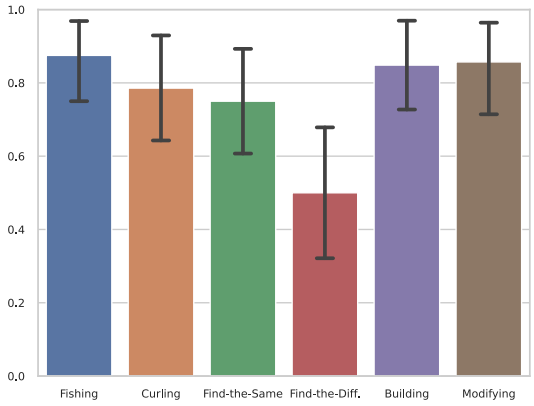

Figure 3: **Success rate (vertical) on 6 goal specifications (horizontal) in the goal-specified experiment ($n = 75$ independent participants).**

directly from raw data: (1) the number of objects within task abstraction, echoing the number of arguments; (2) the distribution of interactions to the objects within task abstraction, describing the emergent semantics (roles) of different objects; (3) distribution of directions of relations within task abstraction, testing the directionality over relations; (4) distribution of absolute paces of the objects within task abstraction, probing the overall intensity of observation and intervention; (5) spatial distributions of interventions, categorizing the patterns of intervention. See Tab. 2 for details.

Results

The convergence of task abstraction

We test the prerequisites of experimental settings to maintain the validity of our data analysis. First, we find no evidence supporting a significant difference in human performance in regard to success rate across the three families of goals ($\chi^2(2) = 0.539, P = 0.463$) and within the families ($\chi^2(1) = 0.104, P = 0.747$), see Fig. 3. Second, to keep our probe of prior on task specification isolated, we test whether there is any learning during a participant's solving of the three tasks. According to the success rate of three tasks coming with different sequences, no evidence shows that people transfer knowledge or strategy acquired from precedent tasks to subsequent ones ($\chi^2(1) = 2.5, P = 0.592$). These suggest that our paradigm is robust to the variance of experimental conditions.

We fit the hierarchical model in human data. The significance of the prior model $p(\theta|t)p(t)$ is tested by the posterior $p(t,\theta|\pi)$. To have a closer look into the feature space, we first calculate the number of clusters on each dimension (see Tab. 2). On Tool Use, Discovery, and Crafting, all the features are tested significant. See Fig. 4A for visualizations of the clusters.

Interestingly, we find that in different variants of the same family of goal specification, task abstractions are well converged—inner-family differences are significantly smaller than inter-family differences ($\chi^2(5) = 28.257, P < 0.0005$). Even Fishing and Curling in Tool Use, where the two goal specifications seem to be very different—the former aims to fetch, while the latter is to hit—but these two goals are both visible in advance and need to achieve beyond capacity. Thus, the task abstraction is constructed as something resembling tool use, with some significant features such as the intervention is clustered at one object—the object is assigned as the role of "tool" in the task abstraction of tool use, and the other one in the inaccessible area is assigned as the role of "target for manipulation". To note, the "one object" here includes all objects that have been tried as a "tool", rather than an instantiated one. Further, the directionality of relations between objects in the task abstraction are intersected at a very small area. The area is around the initial position of the target object in Fishing or in the target area of Curling. This echoes the task-oriented nature of tool use, based on the affordance of the environment for people (Gibson, 1979) (*i.e.*, stay in the accessible area). In contrast, the two variants in Discovery do not show any cue for the directionality of relations—the directions are uniformly distributed. Also, the interventions are uniformly distributed on all objects in the environment. These distinct Discovery from Tool Use by generally assigning all objects with the same role—resource of queries. Thus, they are treated as a whole group of experimentation or observation. In Discovery, objects are also treated as a whole, but in a global-and-local manner, where objects not only in the environment but also in the store bar are of interest. See Fig. 4B for visualizations of the clusters in the spatial domain.

We also compare model significance between data generated by participants that succeeded in the task and by participants that failed the task. No evidence supports that participants cannot construct similar task abstractions even fail to reach the goal ($\chi^2(5) = 2.982, P = 0.702$). This may suggest that the ability to generate task abstraction with the absence of object-level prior is general across individuals.

The divergence of policy

On the basis of task abstraction, we probe the role of another component of interest—the bias model $p(\pi|\lambda)p(\lambda)$. We aim to test a critical assumption of our model that λ and θ are independent of each other. To ground the generated policies, we project the feature space toward trajectory-level indicators—features described by a medium number of clusters are selected, while features described by only one cluster or a large number of clusters are discarded[6]. Thus, each family is grounded with one indicator. For Tool Use, intervention with the tool in either of the two accessible areas, "near to starting point but far to target" or "far to starting point but near to target" identifies two groups of greedy and non-greedy policies; for Discovery, the number of objects involved into one observation, "check interaction one-by-one"

[6] This means that current feature may not be interesting to trajectories in target family of goal specification (Steyvers & Griffiths, 2007).

Table 2: **The grounded feature of three families of goal specification.** In each cell, the content at the left is the hypothesis made by our model, and the content at the right in the parentheses are parameters of clusters fitted by human data ($n = 75$). U stands for uniform distribution. The names of the features are abbreviated; see text for full names. What in parentheses are the mean of the distribution and its confidence interval at the 95% confidence level.

	num. objs.	dist. interact.	dist. direct.	dist. pace	dist. spatial
Tool Use	2 (1.873 ± 0.151)	on tool obj. (1.419 ± 0.137)	directed (1.818 ± 0.076)	tool obj. ≈ 0, other > 0	confined (1.848 ± 0.070)
Discovery	≥ 2 (3.286 ± 0.255)	U (3.493 ± 0.325)	U (1.222 ± 0.089)	all > 0	U (1.0159 ± 0.024)
Crafting	4 (2.603 ± 0.266)	on half objs. (3.541 ± 0.319)	directed (2.118 ± 0.163)	half objs. ≈ 0	confined (4.672 ± 0.481)

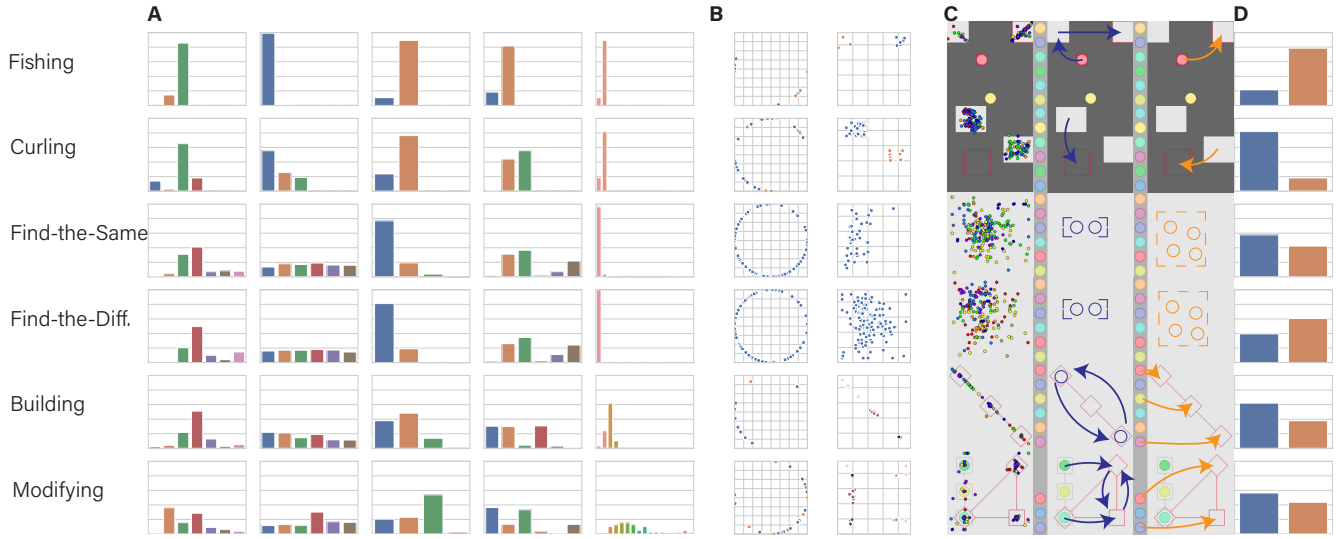

Figure 4: **Results in the goal-specified experiment.** (A) Visualizations of the clusters in 5 features (vertical) on 6 goal specifications (horizontal) in the frequency domain. Except for the axis of num. objs. starts at 0. The rest of the distribution starts at 1. We used data in unit time for num. objs., while other statistics are calculated from the whole sequence. The dist. interact and dist.pace are one-dimensional data, so we directly use statistics to represent them in the distribution of objects within task abstraction. The dist. direct and dist. spatial. of the number of classes after DBSCAN clustering is calculated. (B) Visualizations of the clusters of a subject's sequence in the feature dist. direct. and dist. spatial. (vertical) on 6 goal specifications (horizontal) in the spatial domain. (C) Visualizations of diverse plausible policies. (D) Distribution (vertical) in competing policies (horizontal) on different goal specifications.

(2 objects) and "check a batch of interactions at a time" (≥ 3 objects), categorizes two mindsets of experimenting; for Crafting, the ratio between "adjusting local structure" (*i.e.*, changing the positions of objects within the target shape) and "exchanging material globally" (*i.e.*, replace objects within the target shape with new objects from the store bar) separates two trends of the trade-off between sticking-to-local or switching-to-global. See Fig. 4C for visualizations of the plausible classes of policies. We find that in every task, both in the pair of competing policies emerge (see Fig. 4D)—this may imply the existence of $p(\lambda)$ since the same task abstraction indicates the same task utility, and policies, calculated according to task utility, are also affected by value (preference) (De Martino & Cortese, 2022). Further, we find no evidence showing that the choice between plausible policies and task abstraction are associated ($\chi^2(1) = 1.633, P = 0.201$). This suggests that inner preference is independent of task abstraction. Thus, policies π are generated from both task abstraction θ and inner preference λ.

The effect of constraint

We have observed that people converge on task abstraction and diversify in policies, given goal-specification as task specification. Does constraint also effect task abstraction? We find no evidence supporting the transform of task abstraction by fitting the model of the goal-specified experiment to the data generated by the goal-and-constraint-specified experiment, regardless of constraint family or variant, and also regardless of data source (all trajectories: 89.54% are not significant ($P > 0.05$); all trajectories that succeed: 93.90% are not significant; all trajectories that failed: 85.21% are not significant). Also, specific constraint and goal variants are tested (Easy chances of trial: 93.51% are not significant; Normal chances of trial: 100% are not significant; Hard chances of trial: 93.63% are not significant; Easy steps of action: 93.24% are not significant; Normal steps of action: 96.88% are not

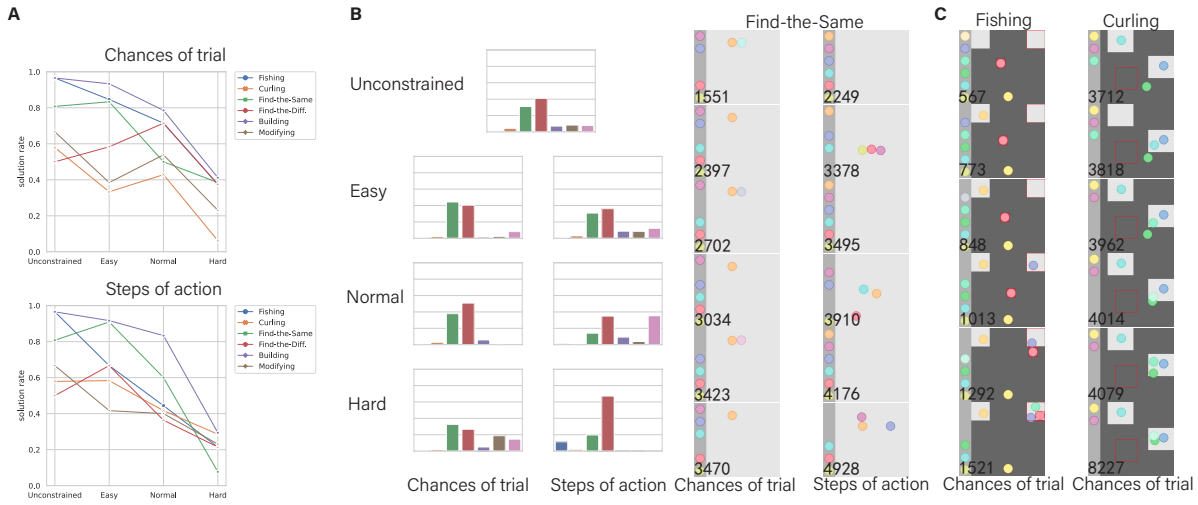

Figure 5: **Results in goal-and-constraint-specified experiment.** (A) Success rate (horizontal right) on different goals (line) with different variants of constraints (horizontal, chances of trial is on the left, steps of action is on the right) ($n = 166$). (B) Case study: in Find-the-Diff, the policy becomes diverse given Hard chances of trial constraint specification, while the policy becomes converged given Hard constraint specification. (C) More showcases of policies under Hard constraint.

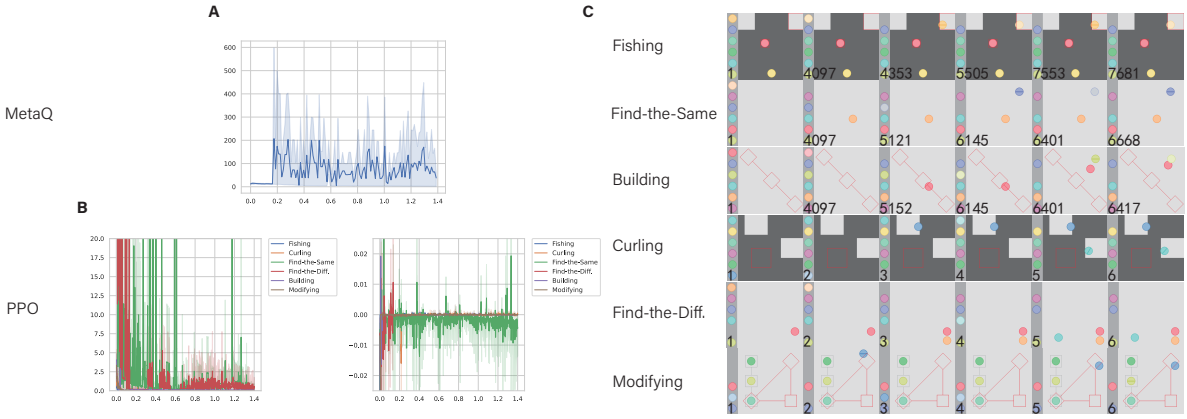

Figure 6: **Results in reinforcement-learning experiment.** (A) Loss function during MetaQ network training. (B) Loss function during PPO network training (loss for value network is on the left, loss for policy network is on the right). Every x-coordinate is stated in terms of 10^7. Three random seeds are chosen, and the average value along with the 95% confidence interval of the loss of the trained agent was presented for each task. (C) Visualization of the trajectories of one MetaQ agent and PPO agent. Each agent exhibits merely a fragment of its play trajectory. Please refer to the appendix for further elaboration (not the same agent).

significant; Hard steps of action: 76.47% are not significant). These may suggest that on the level of task abstraction, constraint specification is subject to goal specification.

As constraint does not make sense independent of goal, the effect of constraints varies according to the goals. Hence, we first find out the goal-constraint pairs that affect performance in regard to success rate, where participants on non-Hard constraints significantly outperform those on Hard constraints. This hypothesis is testified significantly on 6 goal-constraint-specifications. 3 are constrained by steps of action (Fishing: $\chi^2(1) = 4.89, P < 0.05$; Building: $\chi^2(1) = 16.8, P < 0.0005$; Find-the-Same: $\chi^2(1) = 38.4, P < 0.0005$), 3 are constrained by chances of trial (Fishing: $\chi^2(1) = 5.83, P < 0.05$; Curling: $\chi^2(1) = 9.375, P < 0.05$; Building: $\chi^2(1) = 8.93, P < 0.005$; see Fig. 5A). These task specifications are literally harder to solve—participants that succeed are of interest. In these tasks, we find interesting trends that some constraints diversify the policies generated by participants, while some constraints converge the policies. In Find-the-Same, the Hard constraint on chances of trial diverse the policies. Given only 2 chances of trial, participants have to carefully execute experiments to gain enough observations. Thus more objects

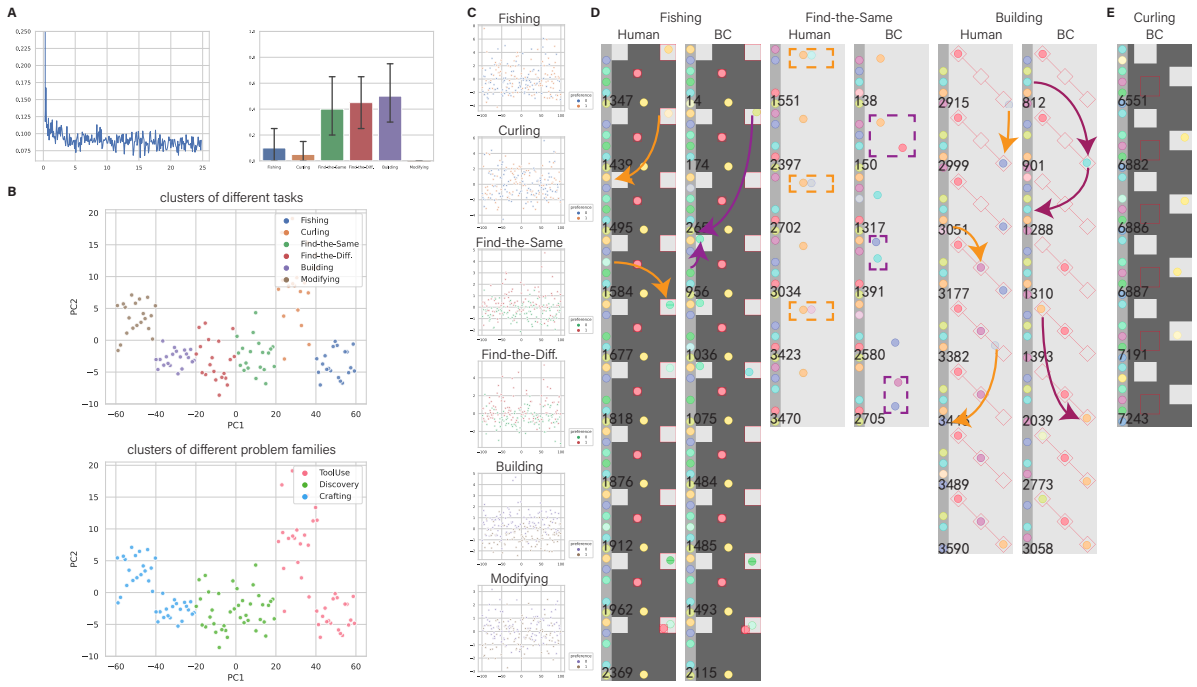

Figure 7: **Results in imitation learning experiment.** (A) Loss function during BC network training (left) and success rate on different goals ($n = 120$; right). (B) Scatter plots of average pooled hidden layer outputs with task label and problem family label, reduced to two dimensions by Principal Components Analysis (PCA). (C) Scatter plots of two-dimensional PCA-reduced average pooled hidden layer outputs with policy label. The policies presented here are identical to the plausible policies depicted in Fig. 4C. (D) Visualization of the trajectories of one BC agent and one subject. (E) Case study: In Curling, the BC agent executes meaningless actions repeatedly, highlighting the deficiencies of our current network architecture design.

are involved, compared with the easier counterparts ($\chi^2(6) = 11.520, P < 0.05$; see Fig. 5B); given only 6 steps of action, participants are just able to place 3 objects into the environment, thus the policy becomes guessing—guess 3 objects in each trial and then check. Both constraints are Hard, but the latter is more strict than the former, for it forms a smaller policy space. A similar trend can also be especially observed in Tool Use: given only one chance of trial; participants have to carefully maintain their tools within accessible areas, for that once the objects are attracted or pushed to the inaccessible area, they fail (see Fig. 5C). These cases echo the classical theories on the trade-off between the utility and complexity of tasks. Since these cases are long-tail distributed, they are not able to be analyzed by statistical approaches. We report the cases here and leave the analysis for future work.

Task abstraction cannot be learned *tabular rasa*

With the ability to generate task abstraction, humans are able to attain our stated goals. Can a gradient learning-based Reinforcement Learning (RL) agent fulfill our goal? MetaQ (Fakoor et al., 2019) and PPO (Schulman et al., 2017; Raffin et al., 2021) are chosen to be taught from scratch in the ProbSol environment (both trained 1.4×10^7 timesteps). The MetaQ algorithm is alternatively trained and regularly adapted to a variety of tasks. We implement action masks

Table 3: **Statistics on the average cumulative reward for RL agents.** This statistic represents the average cumulative reward received by $n = 3$ trained agents (under identical conditions with different random seeds).

	the MetaQ agent	the PPO agent
Fishing	7437.92	7640.08
Cuiling	9752.29	8716.26
Find-the-Same	7317.50	1386.40
Find-the-Diff.	7238.83	1478.94
Building	8653.71	11540.76
Modifying	9543.95	10818.25

(Huang & Ontañón, 2020) in both and switch to semi-Markov Decision Process (semi-MDPs) (Sutton et al., 1999) in MetaQ to improve training outcomes.

The MetaQ agent fails to complete all tasks across all trajectories, but the PPO agent succeeds in the Find-the-Same and Find-the-Difference tasks. See Tab. 3 for the average cumulative reward for RL agents.

The cumulative reward information is intriguing, and the algorithms have converged (see Fig. 6A and Fig. 6B). The success rate, however, indicates that the reinforcement learning agent is incapable of learning or producing the appropri-

ate strategy to complete the task. The MetaQ agent employs a locally optimal strategy for all types of goals, which consists of random actions and a long time between each step (see Fig. 6C). In ToolUse and Crafting, the PPO agent exhibits characteristics similar to random agents. Its accomplishments in Discovery are notable. It executes six consecutive actions before sending out the check action to conclude the game. It does not learn anything but rather remembers the answer based on the gradient. Thus, we conclude that the gradient-learning-based RL cannot generate task abstraction for the ProbSol environment, let alone produce viable strategies.

Alternate approach for analyzing task abstraction

Besides hierarchical modeling, we employ imitation learning to study the properties of task abstraction, to maintain the generality of our analysis. We train an agent by the state-of-the-art Behavior Cloning (BC) algorithm (Gleave et al., 2022) to fit the trajectories generated by human subjects in goal-specified experiments.

BC converges faster than RL and has a higher success rate (see Fig. 7A). From the perspective of trajectory-level policy (with indicators including num.objs., dist. interact. dist. direct., dist. pace, and dist.spatial), we find that the behavior of BC agent is to some extent consistent with that of human subjects (see Fig. 7D). We view the activation vector of the penultimate hidden layer (pooled over the temporal dimension) as the feature of task abstraction extracted by the BC agent. We reduce the dimensionality of the vectors to 2 by PCA into clusters, where each point is one reduced vector activated by one subject-generated trajectory. We find that (i) features in the same task fall in the same cluster via unsupervised clustering (see Fig. 7B); (ii) clusters of different tasks within the same problem family fall into larger clusters (see Fig. 7B), suggesting the existence of shared task abstraction across tasks in the same problem family; (iii) in the same task, features of different policies cannot be separated from each other (see Fig. 7C), demonstrating that policy diversity may be independent of task abstraction. These observations by imitation learning echo our findings by the hierarchical modeling. Unfortunately, we also observe some repetitive actions that humans would never perform (see Fig. 7E) due to the lack of historical memory in the BC agent.

Discussion

In this work, we probed people's ability to generate task abstraction given task specification, with the absence of object-centric priors of semantics, affordance, and physics. We conducted a series of experiments on the ProbSol environment. In goal-specified tasks, we find that task abstractions generated given goal specification are converged regardless of goal variants. Also, we find that policies diverge according to personal preference, which is independent of task abstractions. In goal-and-constraint-specified tasks, we find that task abstractions are generally maintained across different families and variants of constraints. Our hierarchical generative account fits the data well, suggesting that such kind of task abstraction works in a top-down manner, assigning relation- and object-level semantics by grounding the common sense of task specification. The failure of RL *tabular rasa* and the findings in imitation learning both support our interpretation.

This work aims to span a "broad-first" viewpoint that broadly mimics diverse types of problems in the physical world, where each of them needs to be studied in detail. Hence, this work is limited to the "depth-first" perspective. However, as a preliminary work, this work has the potential to lead to several lines of future work: (1) building artificial agents based on the prior of task specification; (2) probing and modeling the exact process of generating tasks abstraction from the general common sense of task specifications; (3) probing and modeling the detailed mechanism of constraint specification that influence task abstraction and policy.

References

Allen, K. R., Smith, K. A., & Tenenbaum, J. B. (2020). Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, *117*(47), 29302–29310.
1, 2, 4

Altman, E. (1999). *Constrained markov decision processes: stochastic modeling*. Routledge.
2

Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., & Girshick, R. (2019). Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, *32*.
2

Battaglia, P. W., Hamrick, J. B., & Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, *110*(45), 18327–18332.
1

Bird, C. D., & Emery, N. J. (2009). Insightful problem solving and creative tool modification by captive nontool-using rooks. *Proceedings of the National Academy of Sciences*, *106*(25), 10370–10375.
2

Blaisdell, A. P., Sawa, K., Leising, K. J., & Waldmann, M. R. (2006). Causal reasoning in rats. *Science*, *311*(5763), 1020–1022.
2

Bransford, J. D., Brown, A. L., Cocking, R. R., et al. (2000). *How people learn* (Vol. 11). Washington, DC: National academy press.
1

Callaway, F., van Opheusden, B., Gul, S., Das, P., Krueger, P. M., Griffiths, T. L., & Lieder, F. (2022). Rational use of cognitive resources in human planning. *Nature Human Behaviour*, *6*(8), 1112–1125.
1

Chu, J., & Schulz, L. E. (2020). Play, curiosity, and cognition. *Annual Review of Developmental Psychology*, *2*, 317–343.

1

De Martino, B., & Cortese, A. (2022). Goals, usefulness and abstraction in value-based choice. *Trends in Cognitive Sciences*.
1, 6

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T., & Efros, A. (2018). Investigating human priors for playing video games. In *International conference on machine learning.*
1

Emde, W., Habel, C. U., & Rollinger, C.-R. (1983). The discovery of the equator or concept driven learning. In *International joint conference on artificial intelligence.*
4

Fakoor, R., Chaudhari, P., Soatto, S., & Smola, A. J. (2019). Meta-q-learning. *arXiv preprint arXiv:1910.00125*.
8

Fan, J. E., Hawkins, R. D., Wu, M., & Goodman, N. D. (2020). Pragmatic inference and visual abstraction enable contextual flexibility during visual communication. *Computational Brain & Behavior*, *3*(1), 86–101.
2

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., ... Taylor, A. (2018). Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data.*
4

Gibson, J. J. (1979). *The ecological approach to visual perception: classic edition*. Psychology Press.
1, 5

Gleave, A., Taufeeque, M., Rocamonde, J., Jenner, E., Wang, S. H., Toyer, S., ... Russell, S. (2022). *imitation: Clean imitation learning implementations.* arXiv:2211.11972v1 [cs.LG]. Retrieved from `https://arxiv.org/abs/2211.11972`
9, 15

Hafri, A., & Firestone, C. (2021). The perception of relations. *Trends in Cognitive Sciences*, *25*(6), 475–492.
4

Hartig, O., & Pérez, J. (2018). Semantics and complexity of graphql. In *Proceedings of the 2018 world wide web conference.*
4

Ho, M. K., Abel, D., Correa, C. G., Littman, M. L., Cohen, J. D., & Griffiths, T. L. (2022). People construct simplified mental representations to plan. *Nature*, *606*(7912), 129–136.
1

Ho, M. K., Saxe, R., & Cushman, F. (2022). Planning with theory of mind. *Trends in Cognitive Sciences*.
1

Huang, S., & Ontañón, S. (2020). A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*.
8

Jara-Ettinger, J., Gweon, H., Schulz, L. E., & Tenenbaum, J. B. (2016). The naïve utility calculus: Computational principles underlying commonsense psychology. *Trends in Cognitive Sciences*, *20*(8), 589–604.
1

Kubricht, J. R., Holyoak, K. J., & Lu, H. (2017). Intuitive physics: Current research and controversies. *Trends in Cognitive Sciences*, *21*(10), 749–759.
1

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.
2

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
2

Osiurak, F., & Badets, A. (2016). Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological Review*, *123*(5), 534.
2

Pelz, M. C., Allen, K. R., Tenenbaum, J. B., & Schulz, L. E. (2022). Foundations of intuitive power analyses in children and adults. *Nature Human Behaviour*, *6*(11), 1557–1568.
1

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1-8. Retrieved from `http://jmlr.org/papers/v22/20-1364.html`
8

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, *497*(7451), 585–590.
1

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
8

Schulz, L. (2012). The origins of inquiry: Inductive inference and exploration in early childhood. *Trends in Cognitive Sciences*, *16*(7), 382–389.
2

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. In *Handbook of latent semantic analysis* (pp. 439–460). Psychology Press.
5

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.
8

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, *331*(6022), 1279–1285.
4

Ullman, T. D., Spelke, E., Battaglia, P., & Tenenbaum, J. B. (2017). Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, *21*(9), 649–665.
1

Ullman, T. D., Stuhlmüller, A., Goodman, N. D., & Tenenbaum, J. B. (2018). Learning physical parameters from dynamic scenes. *Cognitive Psychology*, *104*, 57–82.
2

Zhu, Y., Zhao, Y., & Zhu, S.-C. (2015). Understanding tools: Task-oriented object modeling, learning and recognition. In *Conference on computer vision and pattern recognition.*
1

Implementation details

Reward Shaping

In this subsection, we describe the reward in different problem classes in detail. We define the solution rate of the goals as the agent's reward.

- ***Fishing* in *Tool Use*.** The distance is the minimum normalized Euclidean distance between the target objects $t(o) \in t(O)$ in the environment and all the target areas $t(a) \in t(A)$ when the object target is static.

$$d(s_{-1},g) = \begin{cases} 1, & \\ & s_{-1} = \rho \text{ or } v[t(o)] > \varepsilon, \\ \dfrac{1}{|t(O)|} \displaystyle\sum_{t(o)\in t(O)} \min_{t(a)\in t(A)} \widetilde{L_2}(t(o),t(a)), & \\ & otherwise \end{cases} \tag{4}$$

  where $\varepsilon > 0$ is the threshold for maximum pace of all static objects.

- ***Curling* in *Tool Use*.** The distance is the minimum normalized Euclidean distance between all static objects $o \in O$ in the environment and all the target areas $t(a) \in t(A)$. Let $staticDis(o,t(a))$ equals $\max\{\widetilde{L_2}(o,t(a)), \mathbb{1}(v[o] > \varepsilon)\}$.

$$d(s_{-1},g) = \begin{cases} 1, & \\ & s_{-1} = \rho, \\ \dfrac{1}{|t(A)|} \displaystyle\sum_{t(a)\in t(A)} \min_{o\in O, t(a)\in t(A)} staticDis(o,t(a)), & \\ & otherwise \end{cases} \tag{5}$$

  where $\varepsilon > 0$ is the threshold for maximum pace of all static objects.

- ***Find-the-Diff* and *Find-the-Same* in *Discovery*.** The distance is the minimum normalized Damerau-Levenshtein distance between the id strings of the objects $\langle o|s_{-1} \in O^* \rangle$ in the environment of the last state in the environment and the answer to the task $\langle y|\mathcal{T} \rangle$.

$$d(s_{-1},g) = \begin{cases} 1, & \\ & s_{-1} = \rho \\ \dfrac{1}{\max\limits_{x\in O^*, y\in O^*} lev(x,y)} lev(\langle o|s_{-1} \in O^* \rangle, \langle y|\mathcal{T} \rangle), & \\ & otherwise \end{cases} \tag{6}$$

  Please note that this distance is defined on discrete space. The weights over the three operators add, sub (substitute), and del (delete) are 2, 1, and 2 respectively.

- ***Building* and *Modifying* in *Crafting*.** The distance is the minimum normalized Euclidean distance between all static objects $o \in O$ in the environment and all the block areas $b(a)$ in the target shape $t(S)$.

$$d(s_{-1},g) = \begin{cases} 1, & \\ & s_{-1} = \rho, \\ \dfrac{1}{|t(S)|} \displaystyle\sum_{t(a)\in t(S)} \min_{o\in O, t(a)\in t(S)} staticDis(o,t(a)), & \\ & otherwise \end{cases} \tag{7}$$

  where $\varepsilon > 0$ is the threshold for maximum pace of all static objects.

Reinforcement Learning Training Configurations

In this subsection, we describe the RL algorithm and its training hyperparameters.

Table 4: **The hyperparameters for MetaQ.** The contents on the left of each cell are the MetaQ model's hyperparameters, while the contents on the right are their actual values.

hyperparameters	Value
learning rate	1×10^{-4}
gamma	0.9
replay size	1×10^{6}
snapshot size	3.6×10^{4}
patu	5×10^{-3}
total timesteps	1.4×10^{7}
max path length	1.8×10^{4}
batch size	10240
eps greedy	0.5
eps greedy milestone	5×10^{4}
eps greedy gamma	0.9

Table 5: **The hyperparameters for PPO.** If a hyperparameter is not specified, `sb3_contrib`'s default value is used.

hyperparameters	Value
learning rate	3×10^{-4}
gamma	0.99
total timesteps	1.4×10^{7}
max path length	1.8×10^{4}
batch size	4096

The ProbSol environment is analyzed in order to optimize our RL method. Initially, the finite-state machine (FSM) of actions leads to conditional relationships between actions in action sequences. In fact, a significant proportion of actions at any given time are meaningless and unavailable. Hence, we introduce the action mask (See Fig. 8A and Fig. 8B). `sb3_contrib`[7] gives us with a well-wrapped Maskable-PPO,

[7] `https://github.com/Stable-Baselines-Team/stable-baselines3-contrib`

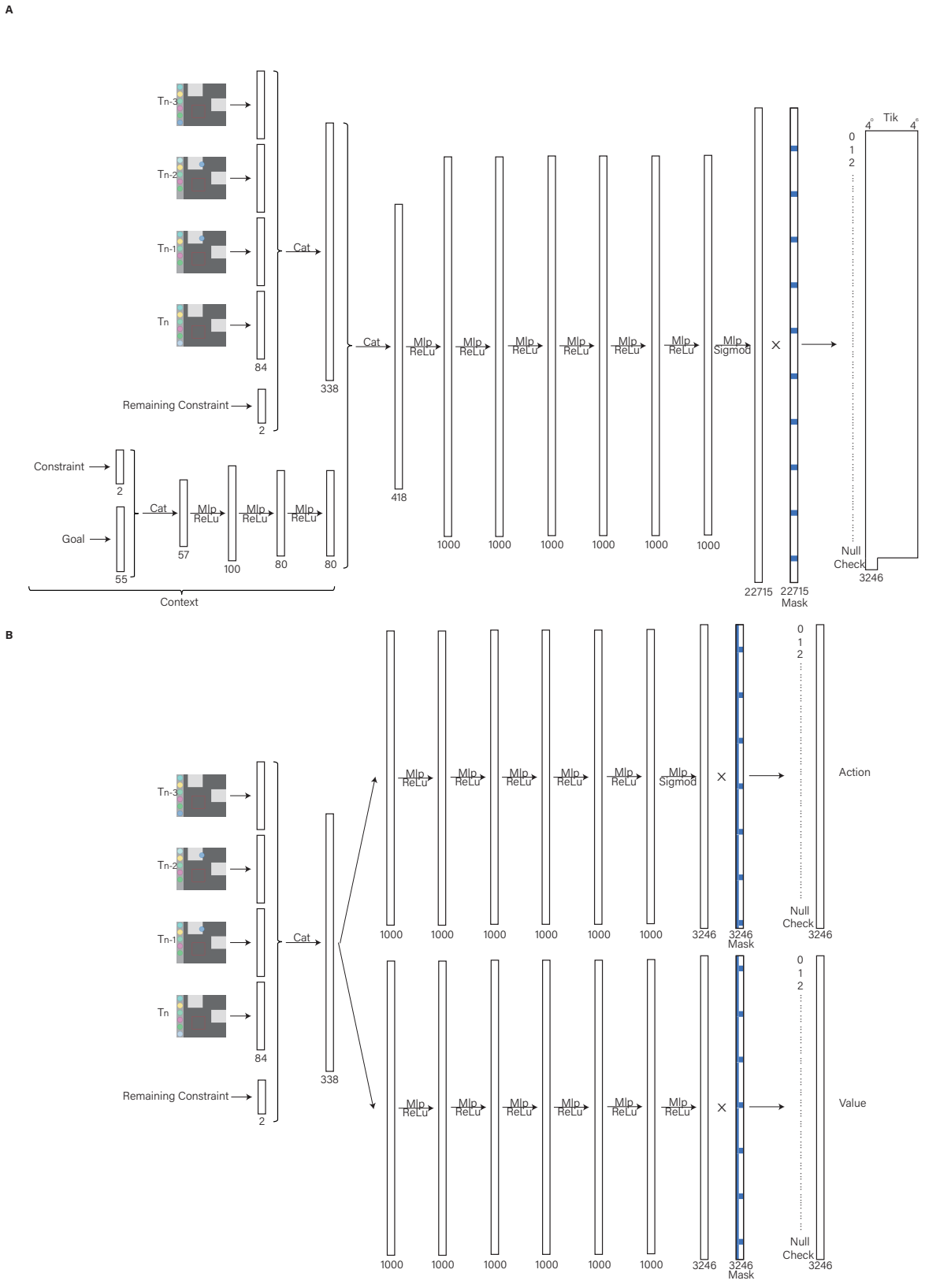

Figure 8: **Architecture of reinforcement learning network.** (A) The value network of the MetaQ algorithm. (B) The policy network and value network of the PPO algorithm.

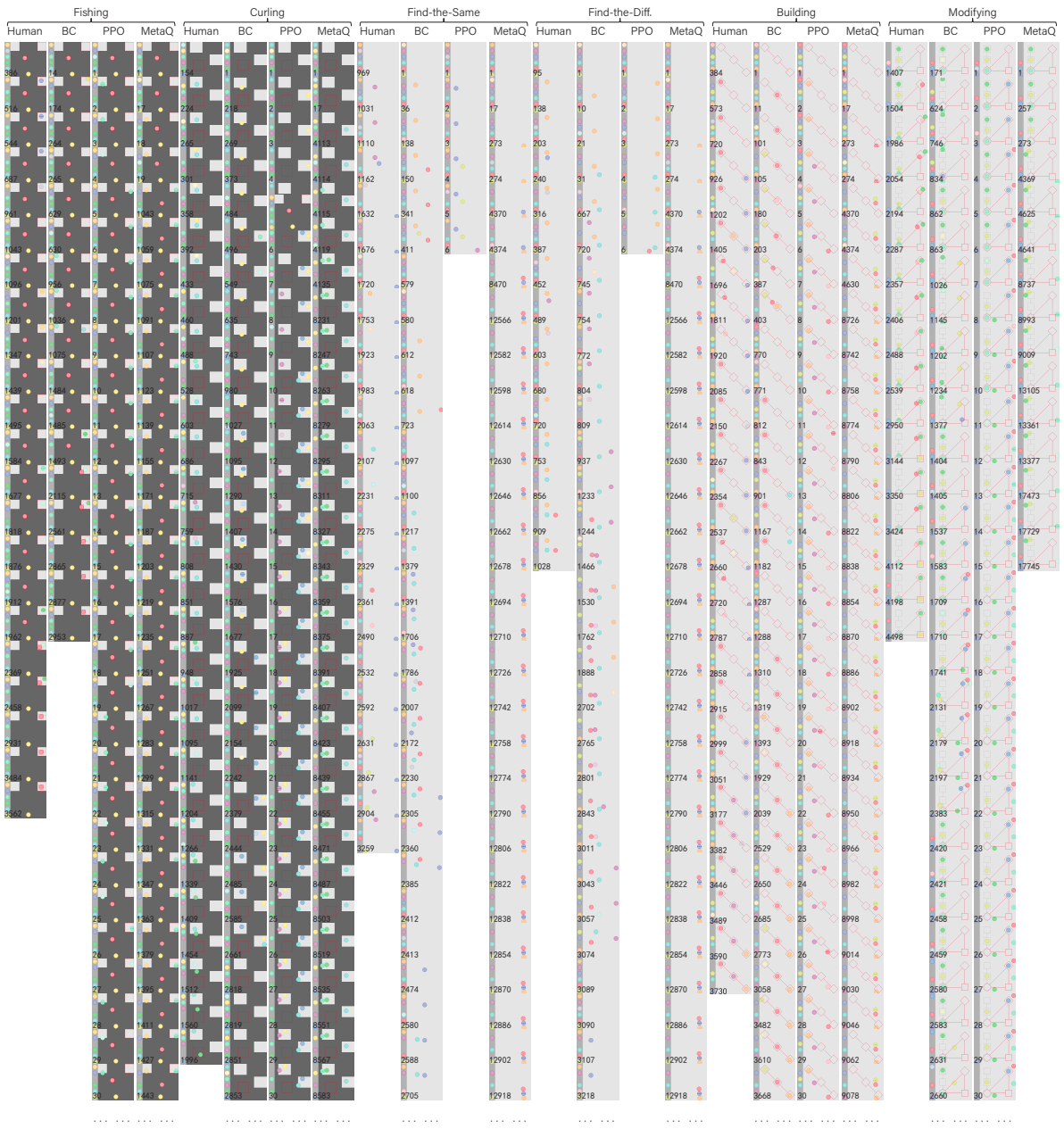

Figure 9: **More trajectories of human, BC and RL agent.** Trajectories of one subject, BC agent, MetaQ agent, and PPO agent are displayed (some trajectories are not fully represented).

so we only need to wrap our environment in the specified format. Secondly, individuals make decisions while observing in ProbSol and proceed once the expected outcome occurs. This resembles semi-MDPs rather than Markov Decision Process (MDPs). Hence, we altered MetaQ to semi-MDPs. The action space contains time information, which indicates that the agent must rest for $n-1$ tik following this action (See Fig. 8A, n comes from 4^0 to 4^6). The MetaQ algorithm is alternately trained on a variety of tasks and regularly adapts to its environment. Thus, we design a context based on the goals and constraints.

See Tab. 4 and Tab. 5 for the MetaQ and PPO hyperparameters we used.

Reinforcement Learning Training Results

In this subsection, we describe the RL algorithm's training result.

See Fig. 9 for trajectory comparison for human and RL agents. In Fig. 6C, We just depict trajectories of one MetaQ and PPO agent accomplishing some goals of the ToolUse, Discovery, and Crafting. Here, we give another MetaQ agent and PPO agent game trajectories as comprehensively as pos-

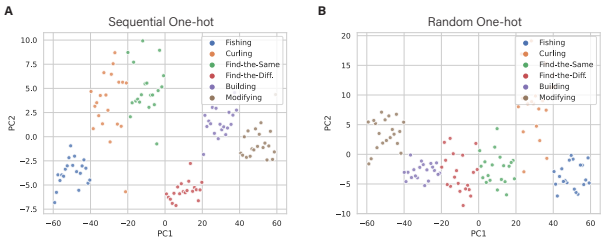

Figure 10: **Support for using one-hot encoding task specifications.** We find no significant differences between the scatter plots of average pooled hidden layer outputs with task labels under different one-hot encoding methods (reduced to two dimensions by PCA). Task specifications are one-hot encoded in training, sequentially (left) or randomly (right).

sible and compare them to the human trajectory, providing more evidence for our conclusion that RL agents cannot construct task abstractions from the environment.

As the results obtained by both algorithms are similar in repeated experiments, we will only qualitatively describe the RL algorithm's feature on the trajectory level. These features highlight the considerable difference between the RL algorithm and humans. Concerning the characteristics in num. objs., dist. interact. and dist. direct., the PPO agent displays an irregular distribution that is similar to the random agent, whereas the MetaQ agent shows a very concentrated distribution. With regards to the dist. pace, the objects used by both agents do not move significantly. As for the feature of dist. spatial, PPO agents tend to take actions near the target region, while MetaQ Agents have very similar location distributions across all tasks. These features are quite inconsistent with the human trajectory shown in Fig. 4A. Therefore, we can infer that RL algorithms do not form appropriate strategies at the strategic level, which is also evident from the success rate.

It is evident that RL algorithms are trapped in local optimality. The resulting policy approximates the greedy one. The MetaQ agent prefers to lazily use the same policy for all tasks (as seen in Fig. 9, where the MetaQ agent's Find-the-Same and Find-the-Diff. tasks take exactly the same trajectory).

We analyze the internal reasons that lead to such different performances between humans and RL algorithms. Firstly, humans can generate task abstractions before solving tasks, while RL algorithms undergo a lot of trial and error in the environment and finally form statistical decisions. Secondly, the guidance for human task-solving comes from a prior and intuition, while the RL algorithm learns from scratch and under the guidance of reward.

Behavior Cloning Training Configurations

In this subsection, we describe the BC data preprocessing and its training hyperparameters.

We employ BC to learn task abstraction from the data collected during our behavioral experiments. Imitation learning (Gleave et al., 2022) is a machine learning approach in which an agent learns to perform tasks by observing and mimicking the behavior of an expert demonstrator. Then, we transformed the problem of training an agent to play the ProbSol game into a supervised learning problem.

Table 6: **The hyperparameters for BC.** The contents on the left of each cell are BC's training hyperparameters, while the contents on the right are their actual values.

hyperparameters	Value
learning rate	4×10^{-3}
L2_regularization	0
epochs	30
batch size	10240

Through preprocessing, 764183 pairs of `(acts, obs, nect_obs, dones, infos)` were obtained from the collected data. First, we identify the successful trajectories from the control group. Second, we map the trajectories' continuous placement space to discrete points in the action space using the nearest-neighbor principle, completing the discretization of actions. Third, we validate whether the discretized action sequences can achieve the desired goal in the environment. Because of the accumulated discretization errors, trajectories could miss their target. Fourth, these sequences are executed in the environment to produce pairs of `(acts, obs, nect_obs, dones, infos)`. Fifth, we augment **obs** with a 6-dimensional one-hot code for the task specification. According to our experiments, various one-hot encoding methods for task specifications do not affect the training results (see Fig. 10). This supports our encoding methods for task specifications.

After acquiring the dataset, we use the `imitation`[8] library for BC, with the same network architecture as the PPO actor's architecture (see Fig. 8B) and hyperparameters shown in Tab. 6.

[8] `https://github.com/HumanCompatibleAI/imitation`