Discovering a symbolic planning language from continuous experience

João Loula (jloula@mit.edu)

Department of Brain and Cognitive Sciences

46 Vassar Street, Cambridge, MA 02139 USA

Tom Silver (tslvr@mit.edu)

Electrical Engineering and Computer Science

32 Vassar Street, Cambridge, MA 02139 USA

Kelsey Allen (krallen@mit.edu)

Department of Brain and Cognitive Sciences

46 Vassar Street, Cambridge, MA 02139 USA

Josh Tenenbaum (jbt@mit.edu)

Department of Brain and Cognitive Sciences

46 Vassar Street, Cambridge, MA 02139 USA

Abstract

Humans make plans with remarkable flexibility by leveraging symbolic representations. How are these representations learned? We present a model that starts out with a language of low-level physical constraints and, by observing expert demonstrations, builds up a library of high-level concepts that afford planning and action understanding. We demonstrate its versatility through experiments inspired by developmental psychology literature.

Keywords: planning; robotics; language of thought; developmental psychology

Introduction

Plans live a double life. On the one hand, they are discrete mental representations: a plan to make yourself a cup of coffee involves creating abstractions like "pick up mug" and "pour coffee in mug". On the other hand, they specify continuous motor actions in the world, like the precise trajectory you will use to reach for the mug, and the angle at which you will pour coffee into it. This separation allows plans to be robust to changes in the environment (like picking up a differently-shaped mug) while still enjoying the compositional benefits of a language of thought (Fodor, 1975) (like recombining abstract procedures such as "pick up X" and "pour X in Y"). This observation has inspired recent work in robotics, which uses pre-built libraries of high-level actions to create plans that are then grounded out in low-level motor commands (Kaelbling & Lozano-Pérez, 2010; Toussaint, 2015; Dantam et al., 2016; Toussaint et al., 2018). This approach allows robots to execute complex sequential plans and use tools in human-like fashion: such models might be a good point of departure to understand how humans implement their abstract plans in the real world.

But there remains the inverse problem—how is it that we develop these symbolic planning abstractions when all we perceive is continuous motion? This is an instance of the symbol grounding problem (Harnad, 1990) that humans solve over and over again: children seem to have the basics of such a language of plans figured out by the time they're a few months old, and can bring it to bear both on executing their own plans (Willatts, 1999) and on understanding others' (Sommerville et al., 2005; Jara-Ettinger et al., 2016; Liu & Spelke, 2017).

In this paper, we propose a model of how an agent, endowed with a simple representation language inspired by core knowledge models of cognition (Spelke & Kinzler, 2007), might learn high-level representations that allow for efficient planning and action understanding. What the model has built-in is the notion of objects, of spatial relations, of agents, and of goals; what it learns are predicates that carve out stable properties of its environment and actions that allow these properties to be changed.

We begin by introducing the planning algorithm of Toussaint et al. (2018), used as a starting point for our model. We then present our approach to inverting that algorithm in order to infer symbolic predicates and actions from expert demonstrations. Finally, we present experiments that showcase how the model draws inferences in different scenarios inspired by cognitive science experiments.

Planning: Hybrid Models

Given an initial state, the model introduced by Toussaint et al. (2018) computes trajectories that reach a goal condition while minimizing a given cost function akin to effort. These trajectories are in general very hard to optimize because of their non-smoothness: objects move in and out of rest, contacts are created and destroyed. Toussaint et al. (2018)'s answer to this is to divide the trajectory into modes, each of which is smooth, and to pre-define a set of actions which allow for specific transitions between modes. Approaches in this spirit are called *hybrid* methods, as the planning occurs in two phases: first the model chooses a sequence of high-level actions, then it computes a low-level implementation of these actions.

More precisely, the model has a pre-specified set of modes s and actions a, and it is given as input an initial state and stable mode x_0 and s_0, a cost function f, and constraints h for the goal, path and mode switches. Its objective is then to solve the following minimization problem:

$$\min_{x,a_{1:K},s_{1:K}} \int_0^T f_{path}(\bar{x}(t))\,dt + f_{goal}(x(T))$$
$$\text{s.t.}\, x(0) = x_0,\, h_{goal}(x(T)) = 0$$
$$\forall t \in [0,T] : h_{path}(\bar{x}(t), s_{k(t)}) = 0,$$
$$\forall k \in \{1,\ldots,K\} : h_{switch}(\hat{x}(t_k), a_k) = 0,$$
$$s_k \in \text{succ}(s_{k-1}, a_k).$$

That is, to find a trajectory x and a sequence of actions $a_{1:K}$ that define a sequence of stable modes $s_{0:K}$ s.t. we can minimize the cost function f over that trajectory, under the mode constraints h_{path}, the action constraints h_{switch}, and the goal constraint h_{goal}. For the remainder of the paper we'll refer to this entire procedure as the *direct* model.

Action Understanding

Our main technical contribution is a model that takes as input a continuous scene and outputs the predicates that describe the stable relations in the scene as well as the actions that govern the transformation of these predicates.

Segmentation

This part of the model is responsible for carving the input trajectory into segments inside of which there exist only smooth dynamics. Take the trajectory in Figure 1 as example: before the gripper touches the orange block, the dynamics in the scene are smooth and can be described by a gradual increase in the gripper's x position, while the two blocks remain at rest. When the gripper touches the orange block, however, there is an abrupt transition in dynamics as the block ceases to be at rest: the new dynamics can be described as the orange block moving horizontally along with the gripper, and the pink block being at rest. It is the job of the segmentation procedure to cut the scene up into these two parts.

More specifically, we frame the problem of segmenting modes as that of time-series segmentation: going back to the direct formulation presented in section the Planning section, our goal here is, from a state sequence $\{x_t\}$, $t \in [1,\ldots T]$, to infer the switch times $\{t_k\}$, $k \in [1,\ldots K]$, that is, the time-points in the trajectory in which a mode switch occurred. This approach to mode segmentation major has the benefit of exploiting temporal information, in contrast to clustering approaches (e.g. Lee et al. (2017)). We use the greedy top-down segmentation algorithm described in Keogh et al. (2004):

$oldSplits \leftarrow [\,]$
while $error > threshold$ **do**
    $error, newSplit \leftarrow \arg\min piecewiseFit(x, oldSplits \cup newSplit)$
    $oldSplits \leftarrow oldSplits \cup newSplit$

The crucial choice here is the function space used for $piecewiseFit$. Since the state in each mode k is governed by a set of constraints $h_{path}(\cdot, s_k(t))$ that belong to some smooth function class $\mathcal{F}$, we must pick a function approximation space $\mathcal{G}$ which we believe to be a good model for $\mathcal{F}$. Since it would be very hard to use the space of all smooth functions, we set $\mathcal{G}$ as a linear function space, which in practice yielded good results, with the only downside of over-segmenting non-linear trajectories. The experiments were run with the value of $1e-5$, though the algorithm proved quite robust to changes in the error threshold. It is also worth noting that nothing precludes the use of more sophisticated functions; such functions could easily be explored in future work.

Predicate discovery and goal inference

Segmentation gives us a set of smooth modes, but these are still continuous: predicate inference is the step that will create discrete, symbolic descriptions for the stable physical properties of these modes. Continuing the example from the previous section, we would like the first segment's stable properties to be described as something like "atRest(■), atRest(■)", and the second one's as atRest(■), hContact(■,■)", where "atRest" is a predicate that specifies that both an object's x and y position are zero, and "hContact" is a predicate that defines a stable horizontal contact between two objects.

Other sets of predicates can also be used to describe the scene, but not all of them are created equal: predicates will be most useful if when they provide a succinct description of each segment's dynamics that is rich enough to support planning. Predicate definitions are not given to the model, however, and at this point in its inference pipeline it has no way of deciding how useful a given set of predicates is—that will be accomplished by the forward planning procedure later. Therefore, all that the predicate inference step does is sample from the derivations of a probabilistic context-free grammar (PCFG) that generates a set of predicates $p_{1:M}$ by combining pre-specified low-level constraints $c_{1:N}$. The probability given by the grammar's derivations is:

$$p(p_{1:M}) \propto \Pi_{m=1}^{M} \gamma \left(\Pi_{n=1}^{N} p(c_n)^{1_{c_n \in p_m}} \right) \quad (1)$$

The model uses the following pre-specified constraints as a basis for building predicates:

- Object1's x velocity or y velocity is equal to 0;
- Object1 and Object2's x position, y position, x velocity or y velocity are equal to each other;
- The difference between Object1 and Object2's x position (resp. y position) is equal to Object1's width (resp. height);
- The difference between Object1 and Object2's x position (resp. y position) is equal to Object1's width (resp. height) or its inverse.

In practice for the experiments we present, the number of constraints present in learned predicates is one or two (see some of predicates learned in our experiments in Table 1): the form of the derivations' probability in equation 1 makes it so

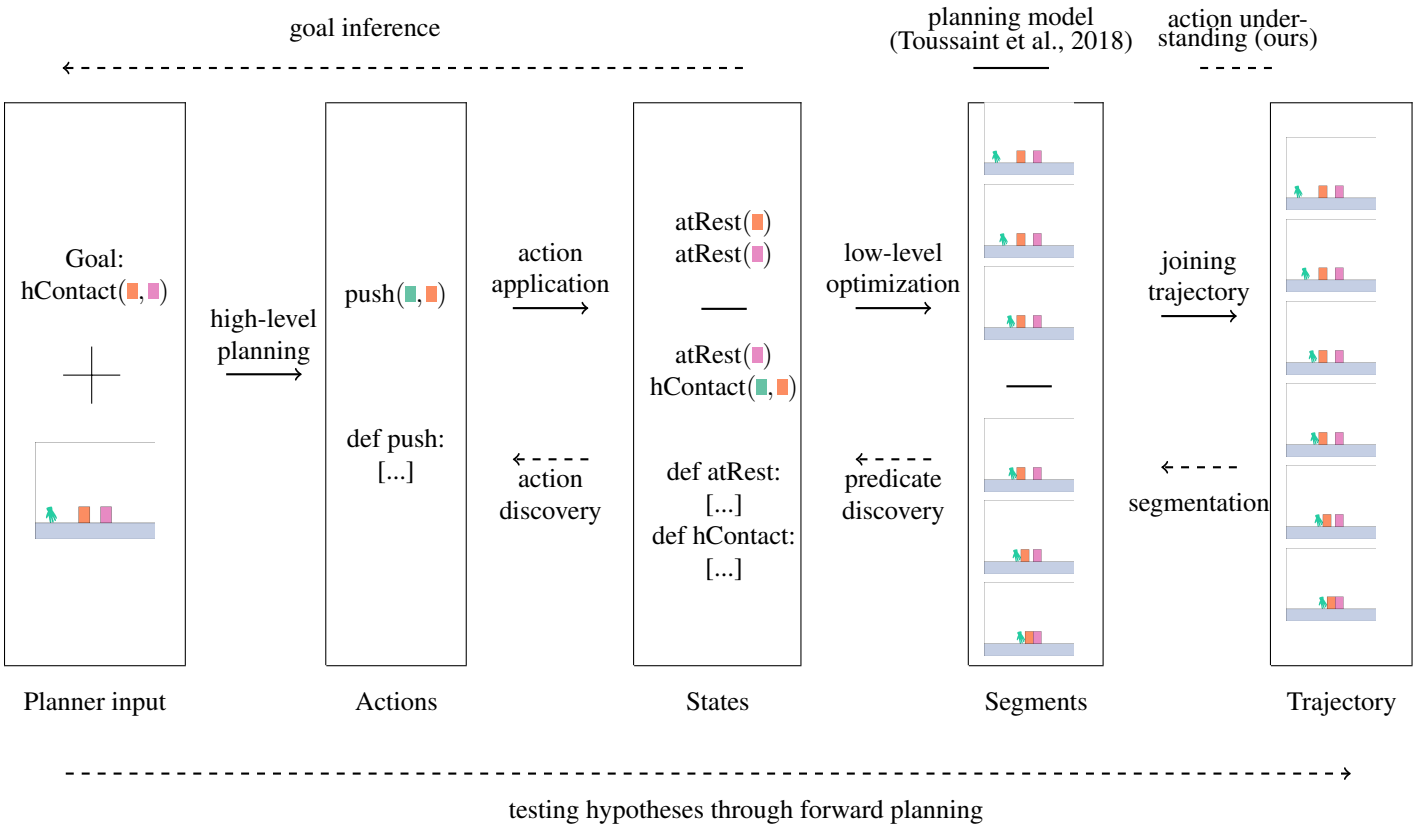

Figure 1: illustration of the planning algorithm of Toussaint et al. (2018) (solid lines) as well as our model, which inverts it (dashed lines). It is important to note that our model is not given the predicates (like "**atRest**") or actions (like "**push**"), and instead has to discover them through experience. Some of the examples of predicates and actions discovered by our model can be found in Table 1.

that the PCFG has a tendency to group constraints together into the same predicate whenever possible rather than split them into different ones.

We will use our proposed set of predicates in order to create a discrete description of each segment in the trajectory. This corresponds to simply applying our list of predicates, under every possible grounding, to each timestep in a segment and checking which ones hold through the entirety of it. **Grounding** refers to the application of predicates to specific objects in the scene, as the predicates themselves are lifted, i.e. abstract. We will call a segment k's **state** the list of grounded predicates that hold through the entirety of that segment, and denote it by s_k. This is also the step in which the goal is inferred, by checking which grounded predicates hold in the final timestep of the trajectory.

Action discovery

After the previous step, we are left with a sequence of discrete states $s_{0:K}$ (lists of grounded predicates), which describe the stable dynamics in each segment of the scene. We will proceed to use these to infer a set of *lifted actions* $\alpha_{1:L}$—transitions between lifted predicates—whose groundings $a_{1:K}$ can explain the transitions between all this sequence of states.

A lifted action α has two components: preconditions and effects, both of which are a set of lifted predicates. We will say that an action grounding a explains a state transition (s_t, s_{t+1}) if s_t contains a's preconditions and the difference between s_{t+1} and s_t is the set of a's effects. If that is the case, we'll say that $succ(s_t, a) = s_{t+1}$.

We can now write the expression for the likelihood of a set of lifted actions given a sequence of states:

$$\begin{aligned} p(\alpha_{1:L}|s_{1:K}) &\propto p(s_{1:K}|\alpha_{1:L})p(\alpha_{1:L}) \\ &\propto p(\alpha_{1:L})\Pi_{k=1}^{K}\mathbb{1}_{\exists a_k \text{ grounding of } \alpha\in\alpha_{1:L},\\ \text{s.t. } succ(s_{k-1},a_k)=s_k}, \end{aligned} \tag{2}$$

that is, we are looking for a sparse set of lifted actions whose groundings can still explain all of the state transitions observed. We pose learning as a program induction problem, where a list of actions is defined by a PCFG (which roughly follows the following generation: action list→action→(preconditions, effects)→predicate list→predicate). Inference is then performed using enumerative search, which in practice outperformed MCMC. This is implemented using the LOTlib library (Piantadosi, 2014).

A subtle aspect of action inference is determining preconditions: in contrast to the classic setting of learning action schemas Drescher (1991), where actions are assumed to take place whenever their preconditions are met, in our problem formulation actions are optional and decided upon by the agent. One consequence of this is that any scene can be explained by actions that have no preconditions at all. Our workaround to this problem is to require that all actions have at least one precondition, and to use the forward planning step as a check on our action inference: if we are too optimistic about when an action might apply, we will likely conceive of plans that seem more efficient than those we observed, which contradicts our rationality assumption.

Testing hypotheses through forward planning

Now that we have built up a library of actions and predicates from the scene, it remains to verify whether these are actually adequate for explaining the expert demonstrations we observed. We will do that by running the planning model in the forward direction given the initial state of the scene, the actions and predicates we learned, and the goal we hypothesized in terms of these learned predicates (which is just the final state, i.e. the set of all grounded predicates that apply to the last timestep in the trajectory). We will then compare the planner's output trajectory with the trajectory we've observed. If they match, we stop our inference and take our hypothesized predicates and actions to be correct. Most of the time, though, we will either fail to conceive of a plan because our learned action set is not rich enough, or we will create a plan that is more efficient than the demonstration we observed—since we are supposing our observations stem from a rational agent, that means our hypothesized predicates are too lax and are underconstraining the scene. In both of these cases, we will go back to predicate inference and sample again from $p(p_{1:M})$.

Experiments

In all the experiments that follow, the model is trained from scratch using as input at train time only a set of trajectories generated by an expert using the algorithm of Toussaint et al. (2018) described in the Planning section. Trajectories are represented as x and y coordinates and velocities for each object in the scene (in the experiments presented here, these are the gripper + one or two blocks, and they maintain their identity across different scenarios in the same experiment). The expert's goal is represented as a predicate (such as hContact(■, ■)) and it remains unchanged between train and test time, but the model does not have access to it: its test time predictions hinge upon its capacity to infer the goal in its correct representation from the train data.

When inferring a trajectory, the model uses the same planning algorithm, but it does not have access to the goal, predicate and actions of the expert, having instead to provide its own learned versions of these concepts. Things that the planning procedure does have baked in and that are used by the model are the notion of objects as cohesive, continuous, solid entities, the spatial layout of the scene, including walls, and the optimizing procedure itself which can be interpreted as a rationality hypothesis.

	Learned predicates	Learned actions

Exp. 2	def **aside**(o1, o2):
	o1.x_position =
	o2.x_position ± o2.width
	def **leftOf**(o1, o2):
	o1.y_position =
	o2.y_position + o2.height
		def **pick**(o1, o2):
		vRest(o1) →
		vContact(o1, o2)

Exp. 3	def **hRest**(o1):
	o1.x_position = 0
	def **hContact**(o1, o2):
	o1.x_position =
	o2.x_position ± o2.width
		pick (as in Exp. 2)
		def **push**(o1, o2):
		hRest(o1) →
		hContact(o1, o2)

Table 1: some of the predicates and actions learned in the experiments. The predicates are written in terms of the pre-specified low-level constraints; the actions are written in terms of the predicates. We have named them according to intuitive interpretations of their role.

Experiment 1: Goal-efficiency

We start with an experiment that mimics the one in Skerry et al. (2013). In that study, 3-month olds observed an adult reaching around a barrier in order to grasp an object (habituation phase). The barrier was then removed, and the infants' looking time was measured as the adult reached for the object either directly or repeating the roundabout motion of the habituation phase, now superfluous. Infants showed surprisal at the ineffective motion, so long as they had been provided with previous experience reaching for objects by means of sticky mittens.

Here, we test whether our model exhibits the same effect by having it observe an expert reach for an object behind a barrier and then analyzing its prediction on a scenario where the barrier has been lifted (Figure 2).

Results

Results for Experiment 1 are presented in Figure 2. We observe that, though many generalizations would have been possible given this single observation, including expecting the exact same trajectory to be performed, the model predicts an optimal path of the gripper towards the object, just as the sticky mittens infants. Two features of the model are crucial for this generalization: (1) the notion of objects that is built

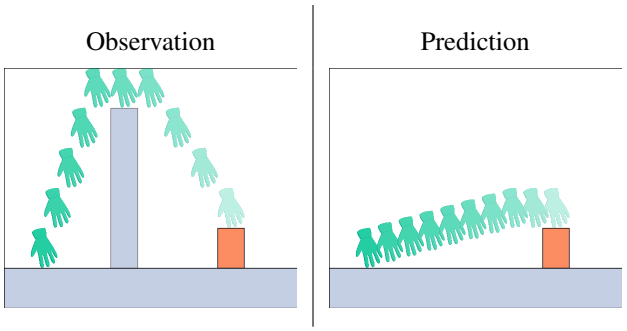

Figure 2 (Experiment 1): the observed expert trajectory (left) and the model's predicted trajectory given a new starting state (right). Different timepoints in the trajectory are superimposed into a single image: lighter objects represent later points in time.

into the model's predicate language, allowing it to infer that doing things with objects is a more interesting goal to people than, say, executing a specific trajectory and (2) the notion of agents as individuals that act rationally to achieve their goals, instantiated in the model by its forward planner pass.

Experiment 2: Conceptual distinctions

In this section, we present an experiment that is analogous to two studies on conceptual distinctions in language and their effect on humans' actions and predictions.

In Hespos & Spelke (2004), infants raised in Korean and English-speaking environments are given a physical prediction task where an object is contained inside another in either a tight or a loose fit. A physical scene then unfolds, whose outcome can either be consistent or inconsistent with the type of fit presented. Crucially, the Korean language has distinct prepositions for tight fit vs. loose fit. The study finds that 5-month old infants from the two groups are surprised when the physical event violates the type of fit presented, but finds that only English-speaking 10-month old infants are insensitive to whether the trial is consistent or inconsistent. The authors interpret the findings as an effect of the environment on the English-speaking babies: though they were initially sensitive to the conceptual distinction, repeated observations grouping the two conditions together make them learn to ignore it.

In Pyers et al. (2010), first- and second-generation Nicaraguan sign language speakers are tested on a navigation task. Second-generation signers distinguish between allocentric and egocentric "left of" and "right of", whereas first-generation signers do not, rendering their concepts of "left of" and "right of" effectively identical to that of "to the side of". When given a navigation task that requires representing the concept "to the left of the blue wall", second-generation signers can successfully solve it, but first-generation signers cannot. Once again, the authors interpret the results as showing that it is the fact that this conceptual distinction is marked in language that allows humans to successfully use it to solve a different task.

Taking inspiration from these studies, we create an experiment where the model's environment (i.e. the set of observations it makes) determines whether it will be sensitive or not to an important conceptual distinction. Two environments are created: in the "Left/right distinction" environment, the model observes an expert pick an orange block up and place it to the left of a pink block in two trials, one of which involves executing a trajectory that is substantially more effortful than placing it to the pink block's right. In the "No distinction" environment, the orange block is placed to the pink block's left or right indiscriminately. The two models are then asked to make a prediction about the same scene.

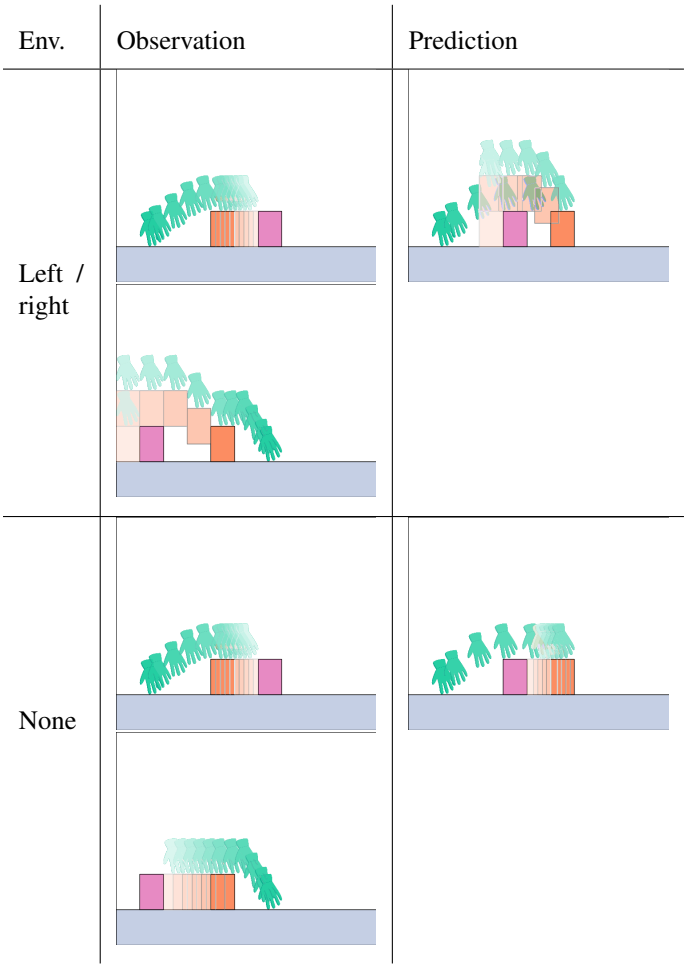

Figure 3 (Experiment 2): two different models are trained from scratch by being given expert trajectories from either the "Left/right distinction" condition (top left) or the "No distinction" condition (bottom left). They are then asked to predict what the expert will do in a new scene: the "Left/right distinction" model's prediction is shown in the top right; the "No distinction" model's prediction is shown in the bottom right. Different timepoints in the trajectory are superimposed into a single image: lighter objects represent later points in time. Trajectory images were downsampled for illustration.

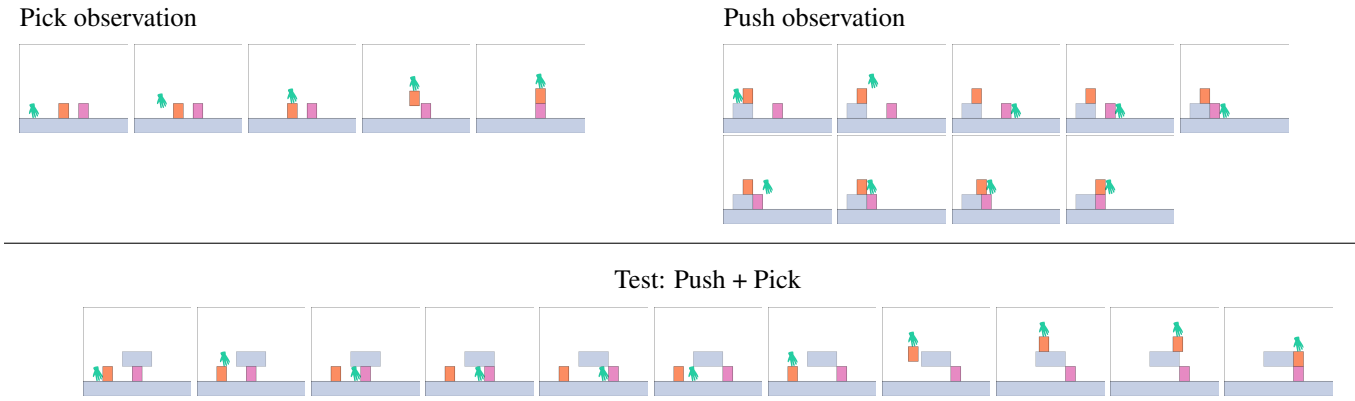

Figure 4 (Experiment 3): the model observes an expert trajectory in a pick demonstration (top left) and a push demonstration (top right), with the same goal. It is then asked to predict the trajectory in a new environment where achieving the goal requires executing both the pick and the push actions: its predicted trajectory is shown at the bottom. Trajectories images were downsampled for illustration.

Results

Results for Experiment 2 are presented in Figure 3. We can see that the two different models make very different predictions for the same scene: the model trained on the "No distinction" environment predicts that the orange block will be simply slid towards the pink one, whereas the model trained on the "Left/right distinction" environment predicts that the orange block will be dragged above the pink one unto its left side. This difference in predictions stems from a difference in learned predicates (see Table 1): akin to the results in Hespos & Spelke (2004) and Pyers et al. (2010), the lack of a distinction between left and right in the environment leads to insensitivity to this distinction in the learned predicates, and vice-versa.

Experiment 3: Compositionality

One of the core features of humans' reasoning power is compositionality: the capacity to create complex concepts by combining simpler ones (Fodor & Pylyshyn, 1988). One way in which compositionality shows up in planning is through the ability to recombine parts of solutions to old problems in order to solve new ones. In our final experiment, the model is trained from scratch separately on two expert demonstrations of the same goal: in the first demonstration an orange block is picked up and placed upon a pink block; in the second demonstration the pink block is pushed towards a wall, and the orange block is then pulled on top of it (top part of Figure 4).

We then combine the learned predicates and actions from the two demonstrations, and ask the resulting model to predict a trajectory in a new environment where successfully placing the orange block atop the pink block requires both pushing and picking actions, thus testing whether it can productively combine two separately learned concepts.

Results

Results for Experiment 3 are presented in Figure 4. We can see that the model not only learns the pick and push actions as well as the relevant predicates (see Table 1), but is also able to recombine them in a novel way in order to predict a complex sequence of actions in a new environment. Such compositional productivity is one of the main claims to fame of actions in hybrid models, and we have shown here that it extends to our learned actions.

Discussion

We have presented a model that starts with a simple language of objects and low-level spatial constraints and, through experience, builds out of it high-level concepts that afford planning and action understanding. We have shown that it comes to understand agents and actions in some of the ways that humans do, such as expecting goal-efficiency and attuning its conceptual vocabulary to its environment, and that it is capable of combining learned concepts in novel ways. Such a model could be useful both for getting robots to learn to create and execute human-like plans as well as to give us insight into how children learn to do this. For instnace, there has been some debate in the literature as to why it is that infants seem to require first-person experience in attaining goals in order to make inferences about goals and acting rationally to achieve them (Sommerville et al., 2005; Skerry et al., 2013). Our model suggests a possible interpretation: though its notion of goals is pre-built, it is still necessary for it to compute an agent's expected trajectory given a certain goal and environment: it could be the case that it is easier to compute this quantity if one has first-person experience performing such trajectories.

The current version of the model suffers from an efficiency bottleneck in that each predicate and action inference proposal requires a forward pass through the planning algorithm

in order to be evaluated. This makes inference very slow as this must be done for hundreds of proposals must be evaluated. It is also a cognitively implausible feature: though a forward pass through a generative model is a standard feature in Bayesian accounts of cognitive tasks, and studies there is some indirect evidence of this happening in action interpretation, it is unlikely that humans would imagine so many different scenarios unfolding in order to interpret a scene. Future work will investigate how to render this inference process more efficient such that it might scale to real-world scenarios.

References

Dantam, N. T., Kingston, Z. K., Chaudhuri, S., & Kavraki, L. E. (2016). Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems* (pp. 1–6).

Drescher, G. L. (1991). *Made-up minds: a constructivist approach to artificial intelligence*. MIT press.

Fodor, J. A. (1975). *The language of thought* (Vol. 5). Harvard University Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*(1-2), 3–71.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, *42*(1-3), 335–346.

Hespos, S., & Spelke, E. (2004). Conceptual precursors to language. *Nature*, *430*(6998), 453.

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.

Kaelbling, L. P., & Lozano-Pérez, T. (2010). Hierarchical planning in the now. In *Workshops at the twenty-fourth aaai conference on artificial intelligence.*

Keogh, E., Chu, S., Hart, D., & Pazzani, M. (2004, June). Segmenting Time Series: A Survey and Novel Approach. In *Series in Machine Perception and Artificial Intelligence* (Vol. 57, pp. 1–21). World Scientific. Retrieved 2018-12-20, from http://www.worldscientific.com/doi/abs/10.1142/9789812565402_0001 doi: 10.1142/9789812565402_0001

Lee, G., Marinho, Z., Johnson, A. M., Gordon, G. J., Srinivasa, S. S., & Mason, M. T. (2017, October). Unsupervised Learning for Nonlinear PieceWise Smooth Hybrid Systems. *arXiv:1710.00440 [cs]*. Retrieved 2018-12-19, from http://arxiv.org/abs/1710.00440 (arXiv: 1710.00440)

Liu, S., & Spelke, E. S. (2017). Six-month-old infants expect agents to minimize the cost of their actions. *Cognition*, *160*, 35–42.

Piantadosi, S. T. (2014). *LOTlib: Learning and Inference in the Language of Thought.* available from https://github.com/piantado/LOTlib.

Pyers, J. E., Shusterman, A., Senghas, A., Spelke, E. S., & Emmorey, K. (2010). Evidence from an emerging sign language reveals that language supports spatial cognition. *Proceedings of the National Academy of Sciences*, *107*(27), 12116–12120.

Skerry, A. E., Carey, S. E., & Spelke, E. S. (2013). First-person action experience reveals sensitivity to action efficiency in prereaching infants. *Proceedings of the National Academy of Sciences*, 201312322.

Sommerville, J. A., Woodward, A. L., & Needham, A. (2005). Action experience alters 3-month-old infants' perception of others' actions. *Cognition*, *96*(1), B1–B11.

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

Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Ijcai* (pp. 1930–1936).

Toussaint, M., Allen, K., Smith, K., & Tenenbaum, J. (2018). Differentiable physics and stable modes for tool-use and manipulation planning. *Proceedings of Robotics: Science and Systems, Pittsburgh, PA*.

Willatts, P. (1999). Development of means–end behavior in young infants: Pulling a support to retrieve a distant object. *Developmental psychology*, *35*(3), 651.
[pages=-]template-graphics[pages=-]template-floats