Learning to act with objects, relations and physics

by

Kelsey Rebecca Allen

Submitted to the Department of Brain and Cognitive Sciences
in partial fulfillment of the requirements for the degrees of

Doctor of Philosophy

and

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . .
Department of Brain and Cognitive Sciences
May 3, 2021

Certified by . . . .
Joshua B. Tenenbaum
Professor, Department of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by . . . .
Rebecca Saxe
Associate Head, Department of Brain and Cognitive Sciences

Learning to act with objects, relations and physics

by

Kelsey Rebecca Allen

Submitted to the Department of Brain and Cognitive Sciences
on May 3, 2021, in partial fulfillment of the
requirements for the degrees of
Doctor of Philosophy
and
Master of Science

Abstract

Humans display an unrivalled degree of control over their environments. From a young age, humans represent the world in ways that allow them to not just make inferences about how the world works, but also to *act* and *intervene* on the world in order to accomplish their goals. Even children can pick up a new skill like "catapulting" from a single demonstration or just a few trials of experience, while it might take a machine agent several hundreds, thousands, or even millions of attempts to master such a skill. For those focused on better understanding these human capabilities, or for those wishing to build more flexible and efficient machines, the computational question is the same: how do people learn and generalize to new problems from just a handful of experiences?

This thesis presents physical problem solving as a window on the flexibility and efficiency of human and machine action. Across two tasks introduced and studied in this thesis, the Gluing Task and the Virtual Tools game, structured action spaces and mental simulation are crucial to explaining human behavior. These action spaces are both *object-oriented* and *relational*, and their representations can be learned with techniques such as deep reinforcement learning or program induction to enable better generalization to new problems. By combining structured action spaces with mental simulation, humans and machines can be *efficient* in the number of actions they require to solve problems and compositionally integrate information gained by trial-and-error experience with information gained by passive observation. Embodied real world experience can additionally affect how much humans rely on mental simulation in physical problem solving. Individuals born with limb differences, like having only one hand, spend significantly more time *thinking* and less time *acting* when faced with a physical puzzle, perhaps reflecting a higher cost of action learned from their everyday experience. Taken together, these results suggest that the flexibility of human physical problem solving stems from the mental simulations people employ when faced with new problems, while the efficiency of human search rests upon appropriately structured action spaces that can be rapidly transformed through minimal trial-and-error experience.

Thesis Supervisor: Joshua B. Tenenbaum
Title: Professor, Department of Brain and Cognitive Sciences

Acknowledgments

It takes a community to raise a child. It similarly takes a community to raise a scientist, and I am so thankful for the community that raised my scientific self.

First and foremost, thank you to Josh Tenenbaum, for his boundless enthusiasm, inspiration, unwavering support, and compassion. I am immeasurably grateful for his mentorship. He embodies so many qualities that I hope to exemplify in my career: dedication to and excellence in teaching, an infectious enthusiasm and ability to inspire, a grand vision for both research and a more inclusive, just academic environment, and most importantly, an understanding that every researcher is also a human, deserving of compassion, respect, and support.

Thank you to my committee members Roger Levy, Josh McDermott and Leslie Kaelbling. Roger, for his scientific precision and for helping me clarify the importance of my research for the cognitive science community. Josh McDermott, who was my first scientific rotation advisor at MIT, and whom I have always enjoyed talking to about the interactions between cognition and perception (especially textures and objects!). Leslie, who I wish I had talked to even more throughout my PhD. She inspired my research directions in AI for the better, and strongly influenced how the focus of my thesis shifted from speech to other forms of action. Someday, I hope to embody Leslie's mentorship style of continuing to hack on code even while teaching and advising students.

Thank you to my other mentors throughout my degree. Peter Battaglia, who gave me such an incredible opportunity to intern with him at DeepMind, and who convinced me that there was much exciting work to be done in connecting cognitive science to deep learning. Marc Toussaint, who influenced my thoughts and expanded my horizons so dramatically as to completely alter my research trajectory in thinking about how to model physical planning. Nancy Kanwisher, while not an official mentor, has always been fascinating to talk to, and I look up to her immensely as an example of an outstanding human. Tamar Makin, who met with me on a whim while visiting MIT, and suggested we work together on investigating how people with limb differences played the games we had been developing at MIT. The collaboration has been one of my all-time favourites.

Thank you to my undergraduate mentors, Ron Caves, Oliver Stelzer-Chilton and Giuseppe Carenini for convincing me to go to graduate school in the first place. Ron for introducing me to a world of research adjacent to physics, where I could happily avoid hardware entirely and instead stare at images of the earth's surface and try to understand them. Oliver Stelzer-Chilton for his tireless support of my brief particle physics career, his invaluable advice, and for supporting me in transitioning to computational neuroscience. Giuseppe Carenini for having enough faith in me, as an undergraduate with no research background in machine learning, to let me pursue an independent research project in his lab. Without these mentors, I would never have made it to graduate school in the first place.

A special thank you to Mark Goldman, who let a random Canadian undergraduate help him develop models of ant foraging, and had the impeccable integrity to go to the trouble of paying me for my remote efforts. Thank you especially for then inviting me to help out at the Woods Hole computational neuroscience summer school. I was truly the luckiest not-yet-first-year graduate student.

For my first years in graduate school in particular, thank you to Natasha Jacques, Meg McCue, and Chen Sun. Your friendship meant a great deal at a time where I was deeply struggling. Natasha and Chen, thank you for being my glorious and free friends, from the glorious and free True North. I wish we spoke even more regularly than we usually do. Thank you to Nils, for both supporting me through my first years at graduate school, and for introducing me to robotics. I will always treasure our conversations about science, politics and the awesome interconnectedness of research.

Thank you to the dear friends I have made in my time here. Maddie Pelz and Jenelle Feather, for being my best pottery buddies, crafting buddies, adventures-to-the-woods buddies, and so much more. Maddie Cusimano and Luke Hewitt, for the years at little canada and for making me a better, more socially aware person. Tselil Schramm and Tynan McAuley, thank you for the evenings watching South Park or Matthew McConaughey, the tasty sourdough pancakes, the board games, and the overall sense of community you helped me feel. Tom Silver, thanks for coming to MIT! The preconditions for friendship were met when we started collaborating, and I expect the effects will be lifelong. Evan Shelhamer, thank you for everything you have done for me. The scale of your support is truly unbounded.

Thank you to the members of McDermott lab. Wiktor Mlynarski, Max Siegel, James Traer and Ray Gonzalez, thank you for providing ample world-weariness and scientific cynicism. Malinda, thank you for our many baking (mis)adventures and walks through the commons.

Thank you to the members of CoCoSci for always providing a scientifically critical sounding board, and for being so diverse in all your interests. There is no other lab that better reflects my own diversity of interests. Thank you particularly to my office-mates over the years: Tobi Gerstenberg, Kevin Smith, Max Siegel, Max Nye and Max Kleiman-Weiner. I have so many fond memories of getting completely derailed from research and instead chatting about politics, scientific identity, and the insanity of deep reinforcement learning. Max KW has always been a fun, challenging adversarial opponent in conversation, and cooks (not over-cooks!) the best food. An extra special shout-out to Kevin Smith. I do not think I could have finished my PhD without him. My time in the lab improved enormously after he joined.

Outside of my office but still in the lab, thank you to Peaks Krafft for being a weird friend who eventually also acknowledged my own weirdness. Ilker Yildirim, thank you for being one of my mentors in the lab, and continuing to talk with me about nonparametric Bayes, intuitive physics, and object representations. Thanks to Mario Belledonne for his tireless contributions to making the lab a more computationally functional place. Eliza Kosoy, thank you for the zany antics and for bringing so much joy to the lab and to me in particular. Your sense of adventure, mischief and humour are unparalleled. Joao Loula, thank you for our weekly “rumble fun” hours during the pandemic, and for all the wonderful conversations and collaborations in the space of physical planning, development, and tool use. I am so excited to see your research trajectory unfolding.

To the undergraduate students who worked with me in the lab, Ulyana Piterbarg, Jessy Lin, Hanul Sky Shin, Robert Chen, Soumya Ram, Kaili Liu, Austin Garrett, and others, thank you for working with me on projects and helping me become a better mentor. I know you will go on to do amazing things.

Outside of MIT, thank you to the friends I made at the Woods Hole summer school and the various conferences over the years. Jessica Hamrick, who has transitioned from friend to mentor, thank you for being a female role model in computational cognitive science. Nicole Rafidi, Dylan Holmes and Leo Casarsa, we met in Woods Hole for a month, and yet our friendship has persisted! I am so grateful. Jimmy Xia, we met at RLDM in Montreal because I was so excited about your work, and now I am so lucky to count you as a friend I can see at both machine learning and cognitive science conferences. Natalia Velez, David Rolnik, Ishita Dasgupta and others, while we were not students at the Woods Hole summer school simultaneously, I thoroughly enjoyed getting to know you while you were students, and am very thankful to still learn from you now. To all those I TA'ed with at the summer school, it was truly one of the highlights of my PhD to have experienced the majesty of Woods Hole with you all each year.

Finally, to my family (Lesley, Doug, Claire and Arden Allen), thank you for always providing me with a safe space. Sometimes that space is quiet, sometimes it is filled with uncontrollable laughter, and sometimes it is a jumble of competitive shouting, but it always feels like home. There were times when I needed that more than I could ever explain, and I know how lucky I am to have that space to come home to. To my grandparents (Terry and Sharon Allen), thank you for your tireless support and for being the best role models I could ask for. Your equal dedication to scientific education, family, leadership, and charitable volunteering laid out a clear path for my idea of what it means to live a good life. I hope my life is as impactful and fulfilling as the example you have given me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans display an unrivalled degree of mastery over their environments. Over the course of a few hundred thousand years, humans have re-shaped the earth so dramatically that the arrival of humans is considered a separate geological era in the development of our planet: the anthropocene [245]. The anthropocene is marked particularly by the overwhelming number of artifacts humans have created to change the landscape around them. This has culminated in the tools and machines we use to build massive cities, fly in planes, or communicate instantly across the world.

Every human culture we know of creates and re-purposes objects to shape the environment in useful ways. And while most of the tools we use have been created and passed down from other people, the spark of innovation to re-purpose objects to serve as tools exists in all of us: we could easily see how to use a rock as a hammer or put an old book under a table leg to stabilize it even if those are not typical uses. These capabilities come so easily to us that we often forget how complex these behaviors are. Despite the universality in people, only a handful of other animals – just a few great apes and birds – use objects in this way, and we tend to think of these as some of the most intelligent behaviors that other species display [197].

On the other hand, machine intelligence is not nearly so adaptive or flexible. Over the last few decades, there have been remarkable advances in the development of general-purpose algorithms to build machines that can solve many different tasks at a level that matches, or even out-performs, humans. This has led to important advances not just in game-playing [156], but also generating human-level linguistic utterances [178], formulating new cosmological equations [38], directing stratospheric balloons to make the internet more accessible [18], and even discovering the structure of proteins to accelerate scientific discovery [193]. But while the algorithms are general, the systems they give rise to are rigid in their behaviors. None of these systems could be applied to problems very different from what they were trained on or designed for, while the humans that created these algorithms are tackling all of these challenges (and many more) within a single human lifetime. Thus, creating intelligence that is as *adaptive* and *general* as human beings remains a grand challenge. We have a unique opportunity now for computational cognitive scientists and machine intelligence researchers to embark on a shared quest to uncover the processes that give rise to the generality of human cognition, and use these insights to develop more adaptive, general-purpose machine intelligence.

1.1 Overview of thesis

In this thesis, I propose that physical problem-solving in humans and machines is fertile ground for making progress on the dual questions of understanding human adaptivity and creating more efficient, flexible machine intelligence. I propose that the key to human flexibility is a rich internal physical model, while the key to human efficiency and generality lies in the structure of the action spaces we bring to bear on new problems.

In Chapter 2 ("Background"), I review how humans represent the physical world in a way that supports not only prediction of how objects will move, but also causal reasoning, multi-modal fusion, and information seeking behavior. Despite this wealth of knowledge about how people represent physical knowledge, it remains unclear how this is recruited for action. There are at least two possible computational accounts stemming from the problem-solving and reinforcement learning communities which I review and point out limitations to. Lying between physics and action is tool use, which has been widely studied across neuroscience, animal cognition, anthropology and cognitive science. I present some of the ideas from the tool cognition community with a focus on the complementary accounts of *affordances* and *mechanical reasoning*.

As a natural starting point to imagining how to build more flexible, general intelligence, Chapter 3 ("Deep Relational Policies") considers how to improve the flexibility of one of the most successful machine learning advances in the past 6 years, deep reinforcement learning. Building on the intuitive physics work of [15] which showed how people make inferences about the stability of towers of blocks, I investigate a minimal change to this paradigm that connects it to action: how should one glue blocks together to *create* stable towers? Traditional deep reinforcement learning [156] completely fails at this task – generalizing across towers of different sizes proves impossible. However, imbuing these deep learning techniques with relational structure, even without physical models, leads to a dramatic improvement in their ability to *generalize* to new sizes of towers, one of the key areas where humans still strongly outperform machine intelligence.

However, this form of generalization is limited – generalizing to different sizes of block towers is impressive, but the physical world is much more vast and varied than block towers. To investigate physical problem-solving more generally, Chapter 4 ("Rapid Physical Problem-Solving with Mental Simulation") introduces the Virtual Tools Game. The Virtual Tools game was designed with generality in mind: it closely reflects the physical reasoning processes underlying tool use and tool creation across a wide variety of physical concepts such as "catapulting", "supporting" and "tipping". The environment allows for a direct comparison between Artificial Intelligence (AI) approaches and humans, as well as providing vastly expanded data collection capabilities relative to traditional in-person tool use experiments for cognitive scientists. I introduce a model ("Sample, Simulate, Update") that combines structured, object-oriented action priors with physical simulation and a rapid update mechanism to explain human performance in two experiments with the Virtual Tools Game.

In Chapter 5 ("Rapid Strategy Learning with Relational Program Policies"), I consider how people learn *across problems* when the problems they encounter share relational structure. I show that people can generalize relational strategy structure to new problems based only on their own trial-and-error experience, and that they can compose these relational strategies with new physical model variables learned via passive observation. I propose an extension of the "Sample, Simulate, Update" model that accounts for these findings by learning relational action priors represented as probabilistic programs.

Chapter 6 ("Meta-Strategy Learning by Embodiment") shows how the Virtual Tools game can be used to ask and answer questions about the effects of embodiment and development on disembodied physical reasoning. Compared to two-handed individuals, children and adults born with one (or even zero) hands learn different meta-strategies for how to reason about physics which carry over into a task where there are no longer embodiment differences between groups. This suggests that the learning and transfer of strategies goes beyond even tasks in the same domain – abstract strategies can cut across the real and virtual worlds.

Looking towards the future and reflecting on other projects throughout the course of my PhD, in Chapter 7 ("Ongoing and Future Directions") I discuss a variety of projects from machine learning, robotics, and development that contribute to our understanding of the computational mechanisms that support action in the physical world. From data-driven ways for discovering relational action priors, to connecting tool use with motor planning, to using the Virtual Tools game to understand cumulative technological culture, I present several exciting collaborations and areas of ongoing work for physical problem-solving as a general research program.

Finally, in Chapter 8 ("Conclusions"), I review the main contributions and results from this thesis, and point towards a promising path for AI, robotics, and computational cognitive science to come together in a shared quest to build more flexible, general intelligence.

Chapter 2

Background

This chapter provides relevant background for the rest of the thesis. I first focus on describing what we know about people's intuitive physical theories, as well as the different computational approaches that have been used to explain human physical inference. From there, I transition to giving a brief overview of different approaches to modeling human decision making, and point out ways in which these views are complementary. Finally, I give a brief overview of tool cognition specifically, to provide motivation for our focus on the problem of tool use in Chapters 4 - 7. This is not meant to be an exhaustive review, but rather to put the proceeding chapters into context within a broader scope of literature than each chapter covers alone.

2.1 Intuitive Physics

Intuitive physics has interested psychologists for several decades. The term was initially used to describe several shortcomings of human physical reasoning across a variety of different tasks. These early studies focused on the problem of prediction – given some initial scenario, what trajectories do people expect objects to follow? In one of the first experiments within this line of research, McCloskey [148] found that people did not reliably assume that a ball would follow a straight path after exiting a curved tube – some participants expected the ball to continue to curve in its trajectory. A number of additional studies found that people make several other errors. Two prominent instances found people cannot reliably describe the ballistic trajectories of objects [102] and they incorrectly believe that objects dropped from a moving carrier will fall straight down [149].

To explain these peculiar findings, a number of theories have been proposed. Perhaps most well known is ‘impetus theory’, so named for its mimicry of pre-Newtonian physical theories based on the Aristotelean notion of ‘impetus’ (the incorrect notion that objects are imparted ‘impetus’ when thrown, and only stop when the ‘impetus’ dissipates) [147, 148, 125]. While these theories do capture many of the errors made by participants, they do not explain the variety of human responses in these physical reasoning tasks – subtle changes to the scenarios, such as changing the curvature of a tube [148] – can affect the kinds of inferences people make.

Alternate theories to ‘impetus’ have also been proposed to account for shortcomings in human physical reasoning: we rely on a set of biased heuristics [176], physical knowledge is hacked together from a set of basic notions about object interactions [95], or we simply misrepresent forces [230]. In all cases, these accounts emphasize that human physical reasoning relies on incorrect representations of physical principles.

2.1.1 Qualitative Physics

Simultaneously to these discoveries in psychology, intuitive physics was taking hold in artificial intelligence [47, 60, 100, 128]. Qualitative physics or Qualitative Process Theory [59] aimed to show how physical inferences could be made from only qualitative rules (like being “to the left of” or “to the right of” instead of ’displaced by -10 units’ or ’displaced by +10 units’). Such approaches were applied to a variety of physical reasoning domains from liquids [44] to boxes [45] and even to containment [46]. These qualitative models also became popular for explaining analogy formation more generally in different relational settings [73].

Because the relationships between physical or spatial variables are qualitative, they naturally afford abstraction and generalization (see Figure 2-1 for an example). [61] argues that this kind of reasoning is the foundation for human physical concepts: we can identify particular relational rules from particular scenes, and then generate causal structures based

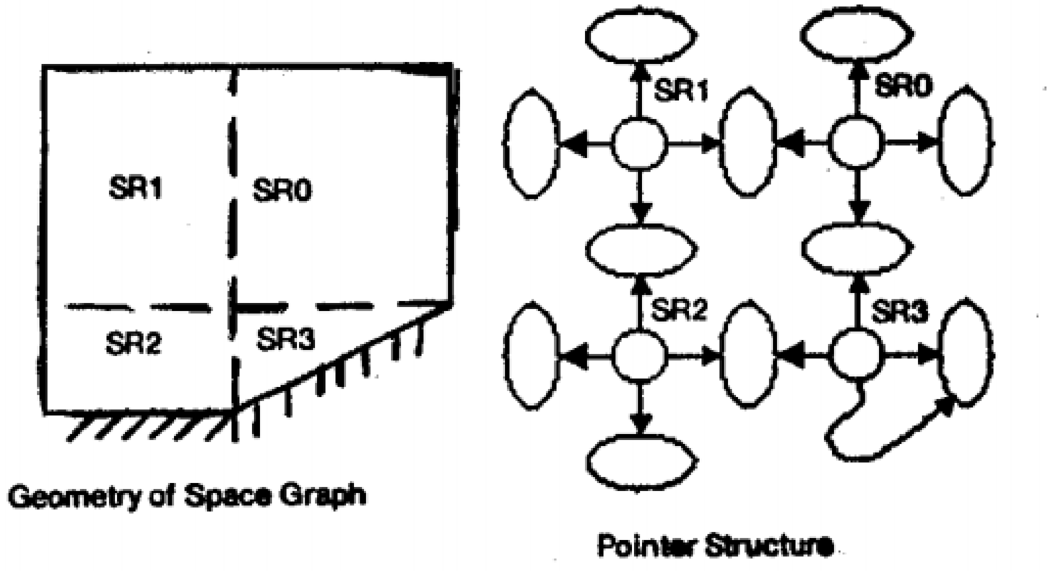


Fig. 2. Space Graph for a scene

The free space in the diagram is broken up into regions in a way that simplifies the description of the kinds of motion possible. The labels on the pointers which indicate the spatial relationships between the nodes are not shown due to lack of space.

Figure 2-1: The FROB reasoning system [57, 58] which breaks up continuous space into discrete regions, keeping track of the qualitative relationships between those regions in a spatial scene graph.

on those rules. These can then be abstracted across different scenes by creating a structural mapping from one scene to another – a process generally referred to as analogy. However, qualitative physics accounts do not universally explain human physical prediction. While they provide good accounts for containment reasoning [46] and question answering in simple scenes [66], they cannot easily explain judgements of kinematics and dynamics, such as whether a ball will have enough momentum to crest a hill [57].

2.1.2 The simulation account

In opposition to these qualitative theories, a separate line of work has suggested that people create mental simulations of mechanical systems that they can use for inference or action [104]. For example, in reasoning about systems of pulleys, people take significantly more

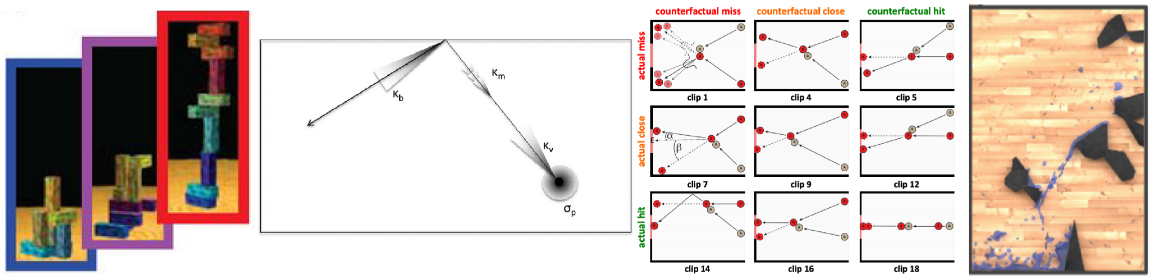


Figure 2-2: A representative set of different experiments investigating the probabilistic simulation account for intuitive physics. Left: examples of the tower stimuli used by [15] to investigate humans' judgements of tower stability. Second from left: diagram of the sources of uncertainty considered by [202] to explain probabilistic trajectory judgements by humans in a bouncing ball task. Second from right: examples of how counterfactual simulations can be used to evaluate whether one ball "almost" caused another one to enter a goal, or to miss a goal (from [77]). Right: Example stimuli for investigating human fluid prediction by [13].

time to make inferences about how pulleys will move when they are further along the causal chain of events from the rope being pulled [103], suggesting that humans do not reason about components independently, but rather simulate a full causal chain of events from an initial action to make inferences. Further lending support for mental simulation and the role of mental imagery, [191] found that people's reaction times to predicting the rate of rotation for interlocking gears depended on the angle of rotation which the gears were offset by. This effect depended on the gears being represented visually – using a schematic of the gears rather than photos nullified the effect.

By themselves, these simulation theories do not obviously give explanations for humans' physical reasoning errors from the previous section. However, by imagining that humans may have uncertainty about how mechanical systems behave, and then by including that in the representation of mental simulations, errors can again arise in how well people can make inferences about physics or mechanical systems more generally. This is often termed the "noisy Newton" hypothesis [183]. It suggests that physical knowledge is instantiated by noisy physical simulation with Newtonian physics [15, 183, 219]. Proponents of a probabilistic simulation based account of physical reasoning suggest that people use Bayesian inference over noisy simulations to perform a variety of physical inference tasks from determining the mass of objects in a scene [97], to predicting how a scene will unfold [15, 79], to making causal physical judgements [77].

Since probabilistic simulation theories rely on Bayesian inference, *uncertainty* and *priors* play a major role in explaining the kinds of errors people make relative to "ground truth" physics. In some of the earliest work on simulation based accounts of physical reasoning, [15] looked at how people made inferences about whether a tower of blocks was stable or unstable (see Figure 2-2 (left)). They demonstrated that perceptual noise (uncertainty about the exact spatial positions of blocks) was critical to explain how people made this inference correctly – their inferences could not be explained by a perfect physical simulator.

[202] showed that perceptual noise alone is not enough to explain people's inferences. Smith and colleagues examined people's predictions about the trajectory of a bouncing ball within a 2D container (Figure 2-2 (second from left)). To explain human judgements, they needed to add *dynamic uncertainty* to the *collisions* that a ball experienced along the walls of the container, as well as the initial direction of velocity of the ball.

These accounts can even be applied to fluids. [13] and [127] showed that an "Intuitive Fluids Engine" could describe how people are able to make inferences about which side of a screen more fluid will fall into (Figure 2-2 (right)). [13] modeled this process using smooth particle hydrodynamics (SPH) models with uncertainty over the initial positions of particles in the fluid.

While perceptual and dynamic uncertainty do not seem to be easily reduced with more observations, other sources of uncertainty around the mass and friction of objects in the scene can be updated rapidly with observed information [244, 201, 236], and can be fused with information from other modalities such as touch [242] and sound [80].

Moving from physical variable inference to causality, simulation based accounts also support counterfactual generation, which can then support causal judgements. For example, [77] suggested that people use a noisy simulation engine to imagine how one billiard ball could have affected another (Figure 2-2 (third panel)), and that this predicts whether they would say that one billiard ball *caused* another to either go into a goal area or not. [79] showed this is not just behavioral – people also move their eyes in ways that track counterfactual simulations to make causal judgements.

While this is an extensive body of work to support the probabilistic simulation account of intuitive physics, none of these approaches explain how people's internal physical models might influence the ways in which humans *act* or *intervene* on the world in order to accomplish their goals. Combining simulation and action is challenging because the number of actions one might consider for realistic problems involving physics, like *constructing* one of the towers from Figure 2-2A, is enormous. If simulations are costly (as suggested by [98]), then simple naive search under a physical simulator would be infeasible for any computationally bounded agent. Understanding how internal physical models link up with action seems to therefore be an important, understudied problem.

Supporting a tighter link between intuitive physics and action, the brain regions involved in physical inference in humans seem to be associated with pre-motor areas [56], suggesting that there may be a tight coupling between internal motor models [234] and internal physical models. These areas have been shown to represent quantitative physical variables such as mass [192], and even tool use apraxia (defined as an inability to use tools) has been associated with lesions to this brain area [170]. We come back to this in the section on tool use below.

2.2 Models in action

Moving away from prediction and towards action, two main bodies of work have been proposed to capture how people solve problems and make decisions.

2.2.1 Problem Solving

In the first line of work, a cohort of computer scientists and cognitive scientists proposed "problem-solving as search" to explain how intelligent agents could make decisions when faced with complex problems. They proposed that problem solving can be thought of as searching through a "problem space" by employing a mental model to understand how states in the space were connected [7, 160]. These ideas were used to explain a variety of phenomena in cognitive science [73, 63, 207], and have recently become popular again for explaining human deductive inference [114] and physical problem-solving [243].

Many of these models rely on a representation of the world that is relational and symbolic. The Stanford Research Institute Problem Solver (STRIPS) has been particularly influential [54]. In STRIPS, *actions*, *scenes* and *goals* are described in the same symbolic language, which allows for a variety of clever planning heuristics that take advantage of this representation [107, 105]. Importantly, actions have symbolic preconditions that describe when they can be applied, and symbolic effects that specify how the world changes when the action is taken.

Task and Motion Planning (TAMP) is an extension of these ideas to the setting in which agents must also make motor plans for how they will execute these symbolic, more abstract actions [116, 139]. In TAMP, problems are broken down into two levels: high-level symbolic reasoning, and low-level motor control. The central similarity across all TAMP methods is that the low-level motor plans that are searched through with TAMP are conditioned on the high-level symbols which are chosen within a plan. For a thorough review of TAMP, please see [69]. Such methods have not generally been directly compared to human action planning; there is significant potential here for explaining how models might integrate with action.

2.2.2 Reinforcement learning

While these structured approaches show much promise to explain how people tackle problems in abstract, symbolic spaces, it is less obvious how they might explain the ways in which people learn, often incrementally, from trial-and-error. To tackle the problem of trial-and-error learning, a wide body of research across neuroscience, psychology and computer science has focused on modeling human decision making using *reinforcement learning* (RL) [210]. Unlike in problem-solving, reinforcement learning makes very few assumptions about the goal of an agent or how the world is described. While problem-solving domains usually specify states and goals in a declarative language, reinforcement learning is a very generic technique that instead assumes an agent is attempting to maximize its *reward*; a scalar quantity that indicates how "good" a particular state for an agent is.

RL problems consist of a set of states in which an agent might find itself, a set of actions which the agent can take, a transition function describing how the environment responds to an agent's actions, and a reward function that defines how good these environment transitions are. The task of an agent is to determine the set of actions to take from a given initial state that will maximize its cumulative reward.

Methods in reinforcement learning are often broadly divided into two types: model-based (MB), and model-free (MF). Model-free methods represent the RL problem an agent faces as simply a collection of states and actions – they are blind to the transition function that governs how the agent's actions affect the environment. They simply represent a set of 'value' estimates that represent a summary of the rewards received based on states and actions the agent has visited in the past. MF methods then use that information to assess the potential benefits of the available actions in the current moment. By contrast, model-based methods include the underlying transition and reward functions that the environment offers in the representation of the problem. Equipped with this extra knowledge, the agent can *plan*, or imagine how a set of states and actions will lead it towards its particular goal. In this way, model-based methods are much closer in spirit and approach to classical problem-solving, but with different reward specifications and associated learning algorithms.

While model-based methods achieve greater flexibility than model-free methods by representing the environment explicitly, model-based methods are also much more computationally costly. In particular, when the state or action space are large, model-based reinforcement learning can suffer from a *combinatorial explosion* in the number of different actions that have to be considered. For this reason, model-free methods are often better descriptions of human behavior after humans have had significant experience with a task, and can presumably summarize their experience more effectively with value functions [75].

An important, but often overlooked, assumption of both types of RL algorithms is that the state and action space are predefined. This is clearly a limitation. When humans and animals learn, these spaces must be discovered, sometimes in parallel with learning how to navigate through them [36, 76, 163, 35]. This has been well documented in the problem-solving community, where 'insights' are largely attributed to a reformulation of the problem space that enables more rapid search [31]. In the reinforcement learning literature, such problem-space discovery has been similarly shown in rodents, where animals can immediately recover responses following extinction [22], and this depends on a reinterpretation of how states are represented [181]. The simultaneous process of task space discovery with task space learning likely involves a wide variety of cognitive constructs, such as categorization, generalization, or causal inference [76, 34, 64].

How task space discovery and learning are done simultaneously remains an important open problem for both the problem-solving and reinforcement learning communities, although some progress has been made. For example, attention can be used to determine which aspects of the environment should be used to represent a state in a way that supports learning [132, 163]. Similarly, humans, animals and machines can learn *hierarchical* structure, often called "hierarchical reinforcement learning" [21, 212].

In hierarchical reinforcement learning, organisms learn abstract hierarchical rules that govern either the structure of the state space (sometimes called "bottlenecks" or "subgoals" [150, 200]), or the structure of the action space (sometimes called "options" [212]), sometimes even when those abstractions are not helpful [36]. Various studies have shown that people do indeed recover bottlenecks or sub-goals when appropriate and can then form abstract actions with these states as the goal [186]. More recently, this has been extended to cases where no obvious bottlenecks exist [237], and where such abstract actions can be composed with model information in discrete grid-world environments when appropriate [64]. Multiple studies have suggested that these state-space abstractions might be acquired by using a mental model of the environment to make inferences about hidden environmentally-dependent state variables that can influence action choices [34, 76], even if the learning algorithm is model-free.

When considering the discovery of structured task spaces jointly with task learning, the line between model-free and model-based approaches thus becomes much murkier; models can be used to modify the way states and actions are represented, even if they are not used to "plan" per se. Similarly, simply making a choice about state and action representations carries assumptions about variables in the environment that matter for making decisions. It is within this murky space that the majority of this thesis work occurs. I emphasize throughout this thesis that the *representations* of actions, states and models is more important for capturing human behavior than the specifics of whether the learning algorithms are model-based or model-free.

2.3 Tool Cognition

Recall the goal of the thesis is to understand how people use internal physical models in order to act and intervene on the world to achieve their goals. There is one very obvious domain in which this is the major topic of study: tool cognition.

Tool cognition has a long history in psychology, neuroscience, anthropology and animal behavior. Anthropologists have studied tools to understand the evolution of the human species, including the suggestion that tool use spurred the development of language [208], as well as linguistic development closely tracking the complexity of stone tools [89, 208]. Cumulative technological culture [216, 23] has long been thought to be of critical importance to the collective success of the human species [48].

Studies of animal behavior have used tool use tasks to measure the intelligence of various species, finding that only a few species of birds (most notably new caledonian crows and the kea) and primates can successfully create, modify, and use tools for new problems [175, 225]. In this field too, reasoning is viewed as central to tool behavior [173, 113]. Developmental psychologists have shown that while even very young children are capable tool users [24, 118], they are unable to make new tools (like hooks) to solve new problems [17, 41] until much later in their development. Finally, a wealth of data in cognitive neuroscience has suggested various brain regions are involved in tool use, with "automatic affordances" of how to grasp an object immediately triggered by the perception of a tool [29, 145].

Clearly then, tool use cuts across many disciplines focused on uncovering the special components that make humans so intelligent. But despite the treasure trove of experimental findings, a computational theory of tool use has remained elusive. In part, this is because tool cognition has been seen differently by different communities. In developmental psychology and animal cognition, tool use is seen as an instance of physical problem-solving: the tasks that animals and children have to perform are unlike anything they would have seen before, and the tools they must make or use are usually novel too [17]. However, the neuroscience and motor learning communities have mainly focused on how *familiar* tools automatically activate hand-specific affordances and memories [25, 20, 213]. The core assumption of this line of work is that tool use is supported by *motor* and *manipulation* knowledge, rather than physical knowledge. As a result, tool representations can be automatically activated (without requiring intention from the organism), and are tied to specific end effectors (like hands).

2.3.1 Affordances

In all communities, affordances have played a central role in shaping how we think about human tool use [167, 213], but these groups differ in how they define and conceptualize affordances. In the ecological approach, initially coined by Gibson [81], an affordance is used to describe what the environment "offers the animal, what it provides or furnishes, either for good or ill". Affordances correspond to action possibilities based on the animal's current action capabilities. Gibson refers to the set of affordances for a particular animal as the ecological niche.

Affordances developed to include notions of intentions [194, 195], where animals do not perceive all affordances in the environment, but rather only those that satisfy a current goal (such as seeing a chair as sit-able when tired, but as cover in case of an earthquake). The perception of affordances is assumed to be *direct* in that it does not require internal representations [195]. Perceiving affordances involves extracting invariant properties of objects, such as seeing "a sharp dihedral angle" as affording "cutting and scraping" [81]. These invariants can also exist relative to the objects being acted on - it is not possible to cut a diamond with a dull knife, but it is possible to cut butter. Critically, it is just this perception of a property (or relative properties of objects) that automatically activates the affordance – no physical model or other internal representation is required. In **Chapter 5**, I present a set of experiments which show that people can learn simplified notions of affordances that are both intentional and relational, as well as a computational model that captures how these might be learned.

2.3.2 Mechanical reasoning

Despite the popularity of the affordance view of tool use, another account has begun to emerge within psychology: that of mechanical reasoning. The reasoning-based approach posits that tool use relies on the ability to form a mental simulation of the required tool-use action, and then "perceive" affordances of objects that suit this intended action. This suggests that tools are mainly represented by the effects they would have in the environment [8], and requires mechanical knowledge corresponding to physical principles [169, 85, 166]. The mechanical knowledge can then guide the selection of appropriate motor actions using an internal motor model [109, 234].

The reasoning view is well aligned with theories in psychology that humans are alone in their ability to "reinterpret the world in terms of hypothetical entities such as causal forces" [173], as well as the wealth of literature alluded to in the Intuitive Physics section of this thesis. Additionally, there are a variety of neuroscientific findings supporting tool use as a mechanical reasoning phenomena. The brain areas associated with tool use (pre-motor areas) overlap significantly with those associated with intuitive physics [56]. Studies of tool use apraxia additionally suggest that people can be selectively impaired in their ability to use novel vs. familiar tools, with deficits in novel tool use associated with lesions to those same pre-motor areas implicated in intuitive physics [166].

In **Chapter 4**, I present a domain for studying this mechanical reasoning aspect of tool use by decoupling tool use from manipulation entirely. I show that a model which uses a combination of *mental simulation* and *action priors* is able to explain how people fine-tune their attempts to solve tasks within this domain, while methods which do not incorporate the model or action priors fall short.

Chapter 3

Deep Relational Policies

While current deep learning systems excel at tasks such as object classification, language processing, and gameplay, few can construct or modify a complex system such as a tower of blocks. One component that these systems lack is a "relational inductive bias": a capacity for reasoning about inter-object relations and making choices over a structured description of a scene. To test this hypothesis, we focus on a task that involves gluing pairs of blocks together to stabilize a tower, and quantify how well humans perform. We then introduce a deep reinforcement learning agent which uses object- and relation-centric scene and policy representations and apply it to the task. Our results show that these structured representations allow the agent to outperform both humans and more naïve approaches, suggesting that relational inductive biases are an important component in solving structured reasoning problems and for building more intelligent, flexible machines.

3.1 Introduction

Human physical reasoning—and cognition more broadly—is rooted in a rich system of knowledge about objects and relations [204] which can be composed to support powerful forms of combinatorial generalization. Analogous to von Humboldt's characterization of the productivity of language as making "infinite use of finite means", objects and relations are the building blocks which help explain how our everyday scene understanding can operate over infinite scenarios. Similarly, people *interact* with everyday scenes by leveraging these same representations. Some of the most striking human behavior is our ubiquitous drive to build things, a capacity for composing objects and parts under relational constraints, which gives rise to our most remarkable achievements, from the pyramids to space stations.

One of the fundamental aims of artificial intelligence (AI) is to be able to interact with the world as robustly and flexibly as people do. We hypothesize that this flexibility is, in part, afforded by what we call *relational inductive bias*. An inductive bias more generally is the set of assumptions of a learning algorithm that leads it to choose one hypothesis over another independent of the observed data. Such assumptions may be encoded in the prior of a Bayesian model [91], or instantiated via architectural assumptions in a neural network. For example, the weight-sharing architecture of a convolutional neural network induces an inductive bias of translational invariance—one we might call a "spatial inductive bias" because it builds in specific assumptions about the spatial structure of the world. Similarly, a relational inductive bias builds in specific assumptions about the *relational* structure of the world.

While logical and probabilistic models naturally contain strong relational inductive biases as a result of propositional and/or causal representations, current state-of-the-art deep reinforcement learning (deep RL) systems rarely use such explicit notions and, as a result, often struggle when faced with structured, combinatorial problems. Consider the "gluing task" in Figure 3-1, which requires gluing pairs of blocks together to cause an otherwise unstable tower to be stable under gravity. Though seemingly simple, this task is not trivial. It requires (1) reasoning about variable numbers and configurations of objects; (2) choosing from variably sized action spaces (depending on which blocks are in contact); and (3) selecting where to apply glue, from a combinatorial number of possibilities. Although this task is fundamentally about physical reasoning, we will show that the most important type of inductive bias for solving it is relational, not physical: the physical knowledge can be learned, but relational knowledge is much more difficult to come by.

We instantiate a relational inductive bias in a deep RL agent via a "graph network", a neural network for relational reasoning whose relatives [185] have proven effective in theoretical computer science [122], quantum chemistry [83], robotic control [226], and learning physical simulation [16, 28]. Our approach contrasts with standard deep learning approaches to physical reasoning, which are often computed holistically over a fixed representation and do not explicitly have a notion of objects or relations [133, 134]. Further, our work focuses on interaction, while much of the work on physical reasoning has focused on the task of prediction [62, 159, 159, 206, 19] or inference [236, 51]. Perhaps the most related works to ours are [134] and [241], which both focus on building towers of blocks. However, while [134]'s approach is learning-based, it does not include a relational inductive bias; similarly, [241]'s approach has a relational inductive bias, but no learning.

This goal of this paper is not to present a precise computational model of how humans solve the gluing task, nor is it to claim state-of-the-art performance on the gluing task. Rather, the goal is to characterize the type of inductive bias that is necessary in general for solving such physical construction tasks. Our work builds on both the broader cognitive literature on relational reasoning using graphs [33, 196, 92, 120] as well as classic approaches like relational reinforcement learning [53], and represents a step forward by showing how relational knowledge can be disentangled from physical knowledge through relational policies approximated by deep neural networks.

The contributions of this work are to: (1) introduce the gluing task, an interactive physical construction problem that requires making decisions about relations among objects; (2) measure human performance in the gluing task; (3) develop a deep RL agent with an object- and relation-centric scene representation and action policy; and (4) demonstrate the importance of relational inductive bias by comparing the performance of our agent with several alternatives, as well as humans, on both the gluing task and several control tasks that isolate different aspects of the full problem.

3.2 The Gluing Task

Participants

We recruited 27 volunteers from within DeepMind. Each participant was treated in accordance with protocols of the UCL Research Ethics Committee, and completed 144 trials over a one-hour session. Two participants did not complete the task within the allotted time and were excluded from analysis, leaving 25 participants total.

Stimuli and Design

The stimuli were towers of blocks similar to those used by [15] and [97]. Towers were created by randomly placing blocks on top of each other, with the following constraints: the tower was constructed in a 2D plane, and each block except the first was stacked on another block. The set of towers was filtered to include only those in which at least one block moved when gravity was applied. In an initial *practice* session, nine unique towers (1 each of 2-10 blocks) were presented in increasing order of size. In the *experimental* session, 135 unique towers (15 each of 2-10 blocks), which were disjoint from the practice set, were presented in a random order in 5 sets of 27. Participants earned points depending on how well they performed the gluing task. They lost one point for each pair of objects they tried to glue, and earned one point for each block that remained unmoved after gravity was applied. As a bonus, if participants used the minimum amount of glue necessary to keep the tower stable, they received 10 additional points. The maximum possible scores in the practice and experimental sessions were 131 points and 1977 points, respectively.

Procedure

Each trial consisted of two phases: the *gluing* phase, and the *gravity* phase. The trial began in the gluing phase, during which a static tower was displayed on the screen for an indefinite amount of time. Participants could click on one object (either a block or the floor) to select it, and then another object to "glue" the two together. Glue was only applied if the two objects were in contact. If glue had already been applied between the two objects, then the glue was removed. Both these actions—applying glue to non-adjacent objects and ungluing an already-glued connection—still cost one point.[1] To finish the gluing phase, participants pressed the "enter" key which triggered the gravity phase, during which gravity was applied for 2s so participants could see which blocks moved from their starting positions. Finally, participants were told how many points they earned and could then press "space" to begin

[1]While this choice of reward structure is perhaps unfair to humans, it provided a fairer comparison to our agents who would otherwise not be incentivized to complete the task quickly.

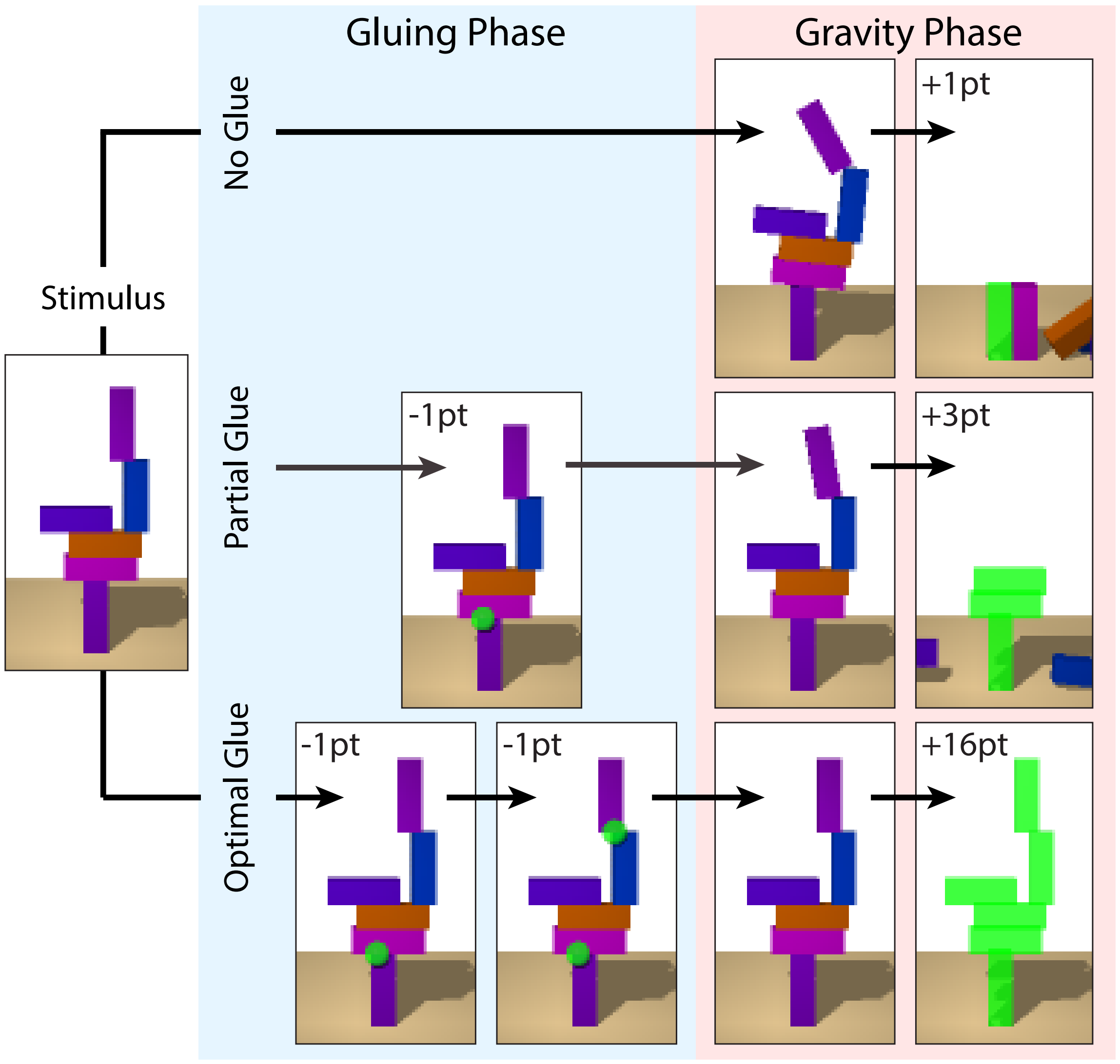


Figure 3-1: **The gluing task.** Given an unstable tower of blocks, the task is to glue pairs of blocks together to keep the tower stable. Three examples of performing the task are shown here. Green blocks in the gravity phase indicate stable blocks. Top: no glue is used, and only one block remains standing (+1 points). Middle row: one glue is used (-1 points), resulting in three blocks standing (+3 points). Bottom row: two glues are used (-2 points), resulting in a stable tower (+6 points); this is the minimal amount of glue to keep the tower stable (+10 points). See `https://goo.gl/f7Ecw8` for a video demonstrating the task.

the next trial. Physics was simulated using the Mujoco physics engine [215] with a timestep of 0.01. After the experiment was completed, participants completed a short survey.

Results

The gluing task was challenging for the human participants, but they still performed far above chance. We discovered several trends in people's behavior, such as working from top-to-bottom and spending more time before applying the first glue than before subsequent glue. The results here represent a preliminary exploration of people's behavior in construction tasks, opening the door for future research and providing a baseline comparison for artificial agents.

Participants achieved an average score of 900 points, with the lowest score being 468 points and the highest score being 1154 points (out of 1977). There was a small (though not significant) effect of learning, with a Pearson correlation of $r = 0.15,\ 95\%\ \text{CI}\ [-0.01, 0.30]$ between trial number and average scaled reward (confidence intervals were computed around the median using 10,000 bootstrap samples with replacement; "scaled rewards" were computed by normalizing rewards such that 0 corresponded to the reward obtained if no actions were taken, and 1 corresponded to the maximum achievable reward).

Participants' response times revealed they were significantly slower to click on the first block in a pair than the second block, with a difference of $t = 4.48\text{s},\ 95\%\ \text{CI}\ [4.34\text{s}, 4.62\text{s}]$. This suggests they had decided on which pair to glue before clicking the first block. People were significantly slower to choose the first gluing action ($t = 4.43\text{s},\ 95\%\ \text{CI}\ [4.30\text{s}, 4.56\text{s}]$; averages computed using the mean of log RTs) than any subsequent gluing action ($t = 2.07\text{s},\ 95\%\ \text{CI}\ [2.00\text{s}, 2.15\text{s}]$; $F(1, 12878) = 149.14$, $p < 0.001$). Also, we found an effect of the number of blocks on response time ($F(1, 12878) = 429.68$, $p < 0.001$) as well as an interaction between the number of blocks and whether the action was the first glue or not ($F(1, 12878) = 14.57$, $p < 0.001$), with the first action requiring more time *per block* than subsequent actions. These results suggest that people may either decide where to place glue before acting, or at least engage in an expensive encoding operation of a useful representation of the stimulus.

On an open-ended strategy question in the post-experiment survey, 10 of 25 participants reported making glue selections top-to-bottom, and another 3 reported sometimes working top-to-bottom and sometimes bottom-to-top. We corroborated this quantitatively by, for

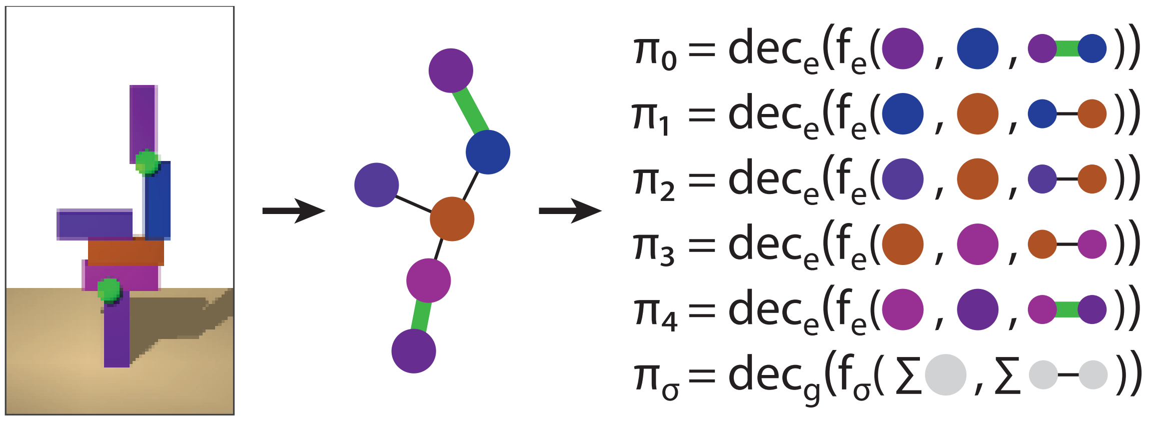


Figure 3-2: **Graph network agent.** First, the positions and orientations of the blocks are encoded as nodes, and the presence of glue is encoded as edges. These representations are then used to compute a Q-value for each edge, as well as a Q-value for taking the "stop" action. See text for details.

each trial, fitting a line between the action number and the height of the glue location, and find their slopes were generally negative ($\beta = -0.07$, 95% CI $[-0.08, -0.06]$).

We compared people's choice of glue configuration to optimal glue configurations, and found that people were significantly more likely to apply glue when it was not necessary (73% of errors) than to fail to apply glue when it was necessary ($N = 3901$, $p < 0.001$). Additionally, participants were very good at avoiding invalid actions: although they had the option for gluing together pairs of blocks that were not in contact, they only did so 1.3% (out of $N = 6454$) of the time. Similarly, participants did not frequently utilize the option to un-glue blocks (0.29% out of $N = 6454$), likely because it incurred a penalty. It is possible that performance would increase if participants were allowed to un-glue blocks without a penalty, enabling them to temporarily use glue as a working memory aid; we leave this as a question for future research.

3.3 Leveraging Relational Representations

What type of knowledge is necessary for solving the gluing task? Physical knowledge is clearly important, but even that implicitly includes a more foundational type of knowledge: that of objects and relations. Inspired by evidence that objects and relations are a core part of human cognition [204], we focus on decomposing the task into a relational reasoning problem which involves computations over pairs of elements and their relations.

3.3.1 Graph Networks

A key feature of our deep RL agent is that it expresses its decision-making policy as a function over an object- and relation-centric state representation, which reflects a strong relational inductive bias. Specifically, inside the agent is a *graph network* (GN), a neural network model which can be trained to approximate functions on graphs. A GN is a generalization of recent neural network approaches for learning physics engines [16, 28], as well as message-passing neural networks [83, 185]. GNs have been shown to be effective at solving classic combinatorial optimization problems [122], inspiring our agent architecture for performing physical construction tasks.

Here, we define a graph as a set of N nodes, E edges, and a global feature G. In the gluing task's "tower graph", nodes correspond to blocks; edges correspond to pairs of blocks; and global properties could correspond to any global piece of information, such as the overall stability of the tower. A GN takes as input a tower graph, and returns a graph with the same size and shape. The representation of the nodes, edges, and globals encode semantic information: the node representation corresponds to position (x) and orientation (q), and the edges to the presence of glue (u). The global features correspond to (or are a function of) the whole graph; for example, this could be the stability of the tower.

Our model architectures first encode the block properties into a distributed node representation $\mathbf{n}_i$ using an encoder, i.e. $\mathbf{n}_i = \text{enc}_n(x_i, q_i; \theta_{\text{enc}_n})$. For an edge $\mathbf{e}_{ij}$, we similarly encode the edge properties into a distributed representation using a different encoder, i.e. $\mathbf{e}_{ij} = \text{enc}_e(u_{ij}; \theta_{\text{enc}_e})$. Initially, the global properties are empty and set to zero, i.e. $\mathbf{g} = 0$. With these node, edge, and global representations, the standard GN computes functions over pairs of nodes (e.g., to determine whether those nodes are in contact)[2], edges (e.g. to determine the force acting on a block), and globals (e.g. to compute overall stability). Specifically, the edge model is computed as: $\mathbf{e}'_{ij} = f_e(\mathbf{n}_i, \mathbf{n}_j, \mathbf{e}_{ij}, \mathbf{g}; \theta_{f_e})$; the node model as

[2]These functions are learned and thus these examples are not *literally* what the agent is computing, but we provide them here to give an intuition for how GNs behave.

$\mathbf{n}'_i = f_n(\mathbf{n}_i, \sum_j \mathbf{e}'_{ij}, \mathbf{g}; \theta_{f_n})$; and the globals model as $\mathbf{g}' = f_g(\mathbf{g}, \sum_i \mathbf{n}'_i, \sum_{i,j} \mathbf{e}'_{ij}; \theta_{f_g})$. The GN can be applied multiple times, recurrently, where $\mathbf{e}'_{ij}$, $\mathbf{n}'_i$, and $\mathbf{g}'$ are fed in as the new $\mathbf{e}_{ij}$, $\mathbf{n}_i$, and $\mathbf{g}$ on the next step.

Applying the GN to compute interaction terms and update the nodes recurrently can be described as *message passing* [83], which propagates information across the graph. In the gluing task, such learned information propagation may parallel the propagation of forces and other constraints over the structure. For intuition, consider the tower in Figure 3-1. After one application of the edge model, the GN should be able to determine which block pairs are locally unstable, such as the top-most block in the figure, and thus require glue. However, it does not have enough information to be able to determine that the bottom-most block in Figure 3-1 also needs to be glued, because it is fully supporting the block above it. Recurrent message-passing allows information about other blocks to be propagated to the bottom-most one, allowing for non-local relations to be reasoned about.

Given the updated edge, node, and global representations, we can decode them into edge-specific predictions, such as Q-values or unnormalized log probabilities (Figure 3-2). For the supervised setting, edges are glued with probability $p_{ij} \propto \text{dec}_e(\mathbf{e}'_{ij}; \theta_{\text{dec}_e})$. For the sequential decision making setting, we decode one action for each edge in the graph ($\pi_{ij} = \text{dec}_e(\mathbf{e}'_{ij}; \theta_{\text{dec}_e})$) plus a “stop” action to end the gluing phase ($\pi_\sigma = \text{dec}_g(\mathbf{g}'; \theta_{\text{dec}_g})$).

3.3.2 Supervised Learning Experiments

Before investigating the full gluing task, we first explored how components of the graph network agent could perform key sub-tasks in a supervised setting, such as predicting stability or inferring which edges should be glued.

To test the GN’s stability predictions, we used towers with variable number of blocks, where the input edges were labeled to indicate whether or not glue was present (**1** for glue, **0** for no glue). Glue was sampled randomly for each scene, and stability was defined as no blocks falling. We tested two settings: *fully connected* graphs (where the graph included all block-to-block edges), and *sparse* graphs (where edges were only present between blocks that were in contact). In both cases, GNs learned to accurately predict the stability of

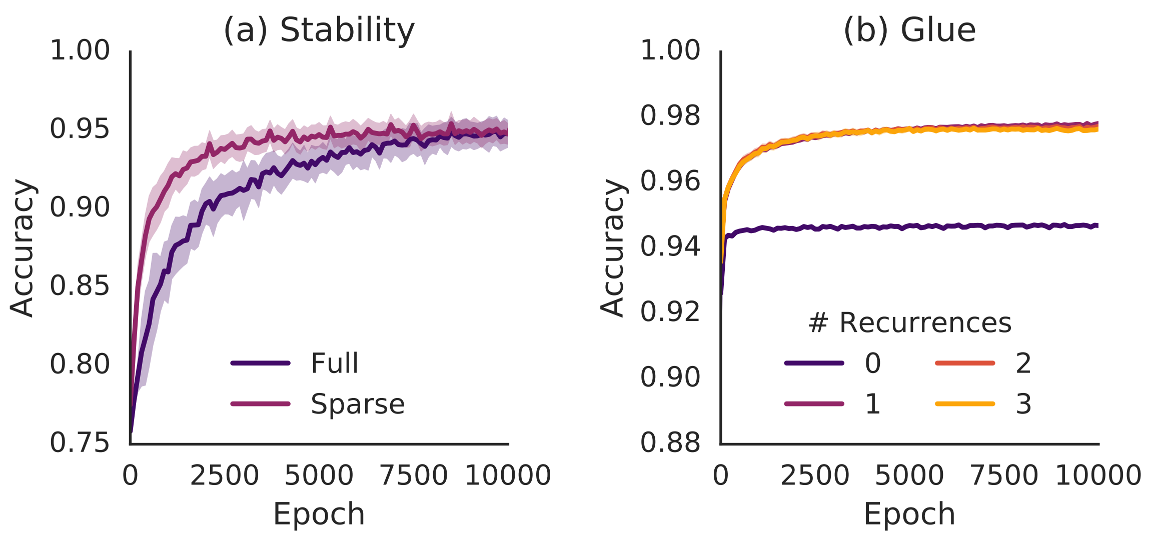


Figure 3-3: **Supervised results for scenes with five blocks.** (a) Stability prediction for input graphs with contact information (sparse) or without (full). (b) Optimal glue prediction for models with different numbers of recurrent steps.

partially glued towers, but the sparse graph inputs yielded more efficient learning (Figure 3-3a). Results are shown for the case of 5 blocks, but these results are also consistent across towers with 6-9 blocks. We also tested whether GNs can learn whether a contact between two blocks should be glued. As discussed previously, some glue locations require reasoning about how forces propagate throughout the structure. We therefore hypothesized that multiple message passing steps would be necessary to propagate this information, and indeed, we found that one recurrence was enough to dramatically improve glue prediction accuracy (Figure 3-3b).

3.3.3 Sequential Decision Making Experiments

From the supervised learning experiments, we concluded that GNs can accurately predict stability and select individual glue points. Next we integrated these components into a full RL agent that performs the same gluing task that people faced, involving multiple actions and delayed rewards.

Design

We considered three agents: the *multilayer perceptron* (or MLP) agent, the *fully-connected graph network* (or GN-FC) agent, the *graph network* (or GN) agent, and the *simulation* agent.[3] As most deep RL agents are implemented either as MLPs or CNNs with no relational structure, our first agent chose actions according to a Q-function approximated by a MLP; as MLPs have a fixed input and output size, we trained a separate MLP for each tower size. The GN and GN-FC agents (which had relational knowledge, but no explicit physical knowledge) also chose actions according to a Q-function and used 3 recurrent steps. The GN agent used a sparse graph structure with edges corresponding to the contact points between the blocks, while the GN-FC used a fully connected graph structure and thus had to learn which edges corresponded to valid actions. Finally, the simulation agent (which had both relational and physical knowledge) chose actions using simulation. Specifically, for each unglued contact point, the agent ran a simulation to compute how many blocks would fall if that point were glued, and then chose the point which resulted in the fewest blocks falling. This procedure was repeated until no blocks fell. Note that the simulation agent is non-optimal as it chooses glue points greedily.

The effect of relational structure

Both the MLP and the GN-FC agents take actions on the fully-connected graph (i.e., they both can choose pairs of blocks which are not adjacent); the main difference between them is that the GN-FC agent has a relational inductive bias while the MLP does not. This relational inductive bias makes a large difference, with the GN-FC agent earning $M = 883.60$, 95% CI $[719.40, 1041.00]$ more points on average (Figure 3-4a) and also achieving more points across different tower sizes (Figure 3-4b).

Giving the correct relational structure in the GN agent further improves performance, with the GN agent achieving $M = 183.20$, 95% CI $[73.20, 302.40]$ more points on average than the GN-FC agent. Thus, although the GN-FC agent does make use of relations, it does not always utilize the correct structure which ends up hurting its performance. Indeed, we

[3] Additional details about the agent architectures and training regimes are available in the appendix.

can observe that the GN-FC agent attempts invalid glue actions—for example, choosing edges between objects that are not adjacent, or self-edges—a whopping 31% (out of $N = 1345$) of the time. The MLP agent similarly picks "invalid" edges 46% (out of $N = 417$) of the time.

The GN agents also exhibit much stronger generalization than the MLP agent. To test generalization, we trained a second set of agents which did not observe towers of 7 or 10 blocks during training, and compared their test performance to our original set of agents. The GN agent exhibited no detectable degradation in performance for either tower size, with a difference in scaled reward of $M = 0.01$, 95% CI $[-0.03, 0.05]$ on 7-block towers and $M = 0.05$, 95% CI $[-0.01, 0.10]$ on 10-block towers. The GN-FC agent interpolated successfully to 7-block towers ($M = -0.04$, 95% CI $[-0.08, 0.00]$), but struggled when extrapolating to 10-block towers ($M = 0.44$, 95% CI $[0.27, 0.61]$). By definition, the MLP agent cannot generalize to new tower sizes because it is trained on each size independently. We attempted to test for generalization anyway by training a single MLP on all towers and using zero-padding in the inputs for smaller towers. However, this version of the MLP agent was unable to solve the task at all, achieving an average of $M = 78.00$, 95% CI $[-140.00, 296.00]$ points *total*.

The effect of physical knowledge

The simulation agent was the only agent which incorporated explicit physical knowledge through its simulations, and we found that it also performed the best out of all the agents. Specifically, the simulation agent earned on average $M = 156.20$, 95% CI $[70.80, 249.60]$ points more than the GN agent, perhaps suggesting that there is a benefit to using a model-based policy rather than a model-free policy (note, however, that the simulation agent has access to a perfect simulator; a more realistic implementation would likely fare somewhat worse). However, we emphasize that the gain in performance by between the GN agent and the simulation agent was much less than that between the MLP and GN-FC agents, suggesting that relational knowledge may be more important than explicit physical knowledge in solving complex physical reasoning problems like the gluing task.

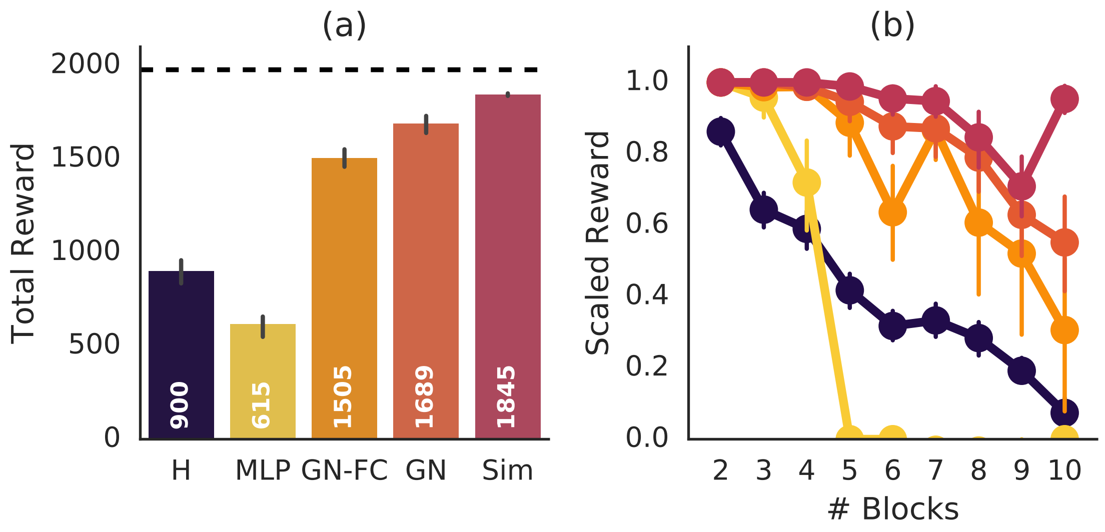


Figure 3-4: (a) Comparison of overall reward for humans and agents. *H*: human; *MLP*: MLP agent; *GN-FC*: GN agent operating over a fully-connected graph; *GN*: GN agent operating over a sparse graph; *Sim*: simulation agent. (b) Comparison of scaled reward across towers of different sizes. Rewards are scaled such that 0 corresponds to the reward obtained when no actions are taken, and 1 to the optimal reward.

Comparison to humans

Although our goal was not to build a model of human cognition on the gluing task, we still compared people's behavior to that of the GN agent to elucidate any obvious differences. Participants' average reward fell between the MLP and GN-FC agents' (Figure 3-4a). As in Figure 3-4b, both agents and humans had increasing difficulty solving the task as a function of tower size, though this was expected: as the number of blocks in the tower increases, there is an exponential increase in the number of possible glue combinations. Specifically, for a tower with k contact points, there are 2^k possible ways glue can be applied (around 1000 possibilities for a 10-block tower), and optimally solving the task would require enumerating each of these possibilities. Our agents do not do this, and it is unlikely that humans do either; therefore, the drop in performance as a function of tower size is not surprising.

Looking more closely, we found the GN agent made different patterns of errors than humans within scenes. For example, while we found that people were more likely to make false positives (applying glue when none was needed), we did not find this to be true of the GN agent (41% of errors, $N = 155$, $p < 0.05$). This difference might be a result of perceptual uncertainty in humans, which leads to a tendency to over-estimate the instability of towers [15].

3.4 Discussion

In this paper, we explored the importance of relational inductive bias in performing interactive physical reasoning tasks. We introduced a novel construction problem—the "gluing task"—which involved gluing pairs of blocks together to stabilize a tower of blocks. Our analysis showed that humans could perform far above chance and discovered they used systematic strategies, such as working top-to-bottom and reasoning about the whole glue configuration, before taking their first action. Drawing on the view from cognitive psychology that humans understand the world in terms of objects and relations [196, 204, 120], we developed a new deep RL agent that uses a decision-making policy based on object- and relation-centric representations, and measured its ability to learn to perform the gluing task. These structured representations were instantiated using *graph networks* (GNs), a family of neural network models that can be trained to approximate functions on graphs. Our experiments showed that an agent with an object- and relation-centric policy could solve the task even better than humans, while an agent without such a *relational inductive bias* performed far worse. This suggests that a bias for acquiring relational knowledge is a key component of physical interaction, and can be effective even without an explicit model of physical dynamics.

Of course, model-based decision-making systems are powerful tools [198], and cognitive psychology work has found evidence that humans use internal physics models for physical prediction [15], inference [97], causal perception [77], and motor control [117]. Indeed, we found that the best performing agent in our task was the "simulation" agent, which used both relational and physical knowledge. Provisioning deep RL agents with joint model-free and model-based strategies inspired by cognitive psychology has proven fruitful in imagination-based decision-making [96], and implementing relational inductive biases in similar systems should afford greater combinatorial generalization over state and action spaces.

More generally, the relational inductive bias possessed by our GN agent is not specific to physical scenes. Indeed, certain aspects of human cognition have previously been studied and modeled in ways that are explicitly relational, such as in analogical reasoning [73, 108]. In other cognitive domains, GNs might help capture how people build cognitive maps of their environments and use them to navigate; how they schedule their day to avoid missing important meetings; or how they decide whom to interact with at a cocktail party. Each of these examples involves a set of entities, locations, or events which participate in interactive relationships and require arbitrarily complex relational reasoning to perform successfully.

In sum, this work demonstrates how deep RL can be improved by adopting relational inductive biases like those in human cognition, and opens new doors for developing formal cognitive models of more complex, interactive human behaviors like physical scene construction and interaction.

3.5 Supplementary Material

3.5.1 Architectural Details

MLP Agent

The MLP agent had three hidden layers with 256 units each and ReLU nonlinearities. Inputs to the MLP agent consisted of the concatenated positions and orientations of all objects in the scene, as well as a one-hot vector of size $E_{fc} = N(N-1)/2$ indicating which objects had glue between them. There were $E_{fc}+1$ outputs in the final layer: one for each pair of blocks plus the floor (including non-adjacent objects), and an additional output for the “stop” action.

GN Agents

Th GN-FC agent had the same inputs and outputs as the MLP agent. The inputs to the GN agent also included the positions and orientations of all objects in the scene, but the “glue” vector instead had size $E_{sparse} \approx N$ (where E_{sparse} is the number of pairs of blocks in contact); the GN agent was also told *which* blocks, specifically, were in contact. There were $E_{sparse} + 1$ outputs in the final layer.

Both GN agents used node, edge, and globals encoders which were each linear layers with an output dimensionality of size 64. The edge, node, and global models were each a MLP with two hidden layers of 64 units (with ReLUs) and an output dimensionality of 64. In these models we also used "skip" connections as in [122], which means that we also fed in both encoded and non-encoded inputs to the model. We additionally used a gated recurrent unit (GRU) as the core for our recurrent loop, similar to [135]. We passed the outputs of the recurrent GN to a second GN decoder (with the same architecture for the edge, node, and global models). This second GN helps the agent decompose the problem, such as first detecting which block pairs are in contact, and then determining which of those pairs should be glued. Finally, the edge and global values were further decoded by two hidden layers of 64 units (with ReLUs) and a final linear layer with a single output.

3.5.2 Training Procedure

Both the GN and MLP agents were trained for 300k episodes on 100k scenes for each tower size (900k total scenes), which were distinct from those in the behavioral experiment. We used Q-learning with experience replay [156] with a replay ratio of 16, a learning rate of 1e-4, a batch size of 16, a discount factor of 0.9, and the Adam optimizer [123]. Epsilon was annealed over 100k environment steps from 1.0 to 0.01.

Because the MLP agent had fixed input and output sizes that depend on the number of blocks in the scene, we trained nine separate MLP agents (one for each tower size). Both GN agents were trained simultaneously on all tower sizes using a curriculum in which we began training on the next size tower (as well as all previous sizes) after every 10k episodes.

Chapter 4

Rapid Physical Problem-Solving with Mental Simulation

In the previous chapter, adding relational inductive biases to a deep reinforcement learning agent was sufficient to allow the agent to generalize to a variety of differently sized block towers. However, the models still required hundreds of thousands of actions during training to find good policies for gluing. Furthermore, while generalizing to different sizes of block towers is impressive, the physical world is substantially more vast and varied than towers of blocks. This chapter introduces both a new domain for studying physical problem-solving more generally (the Virtual Tools game), as well as a new model which relies not just on structured action spaces, but also incorporates a physical simulator to imagine the approximate effects of actions before taking them in the real world. In the Virtual Tools game, people solve a large range of challenging physical puzzles in just a handful of attempts. We propose that the flexibility of human physical problem solving rests on an ability to imagine the effects of hypothesized actions, while the efficiency of human search arises from rich action priors which are updated via observations of the world. We instantiate these components in the "Sample, Simulate, Update" (SSUP) model and show that it captures human performance across 30 levels of the Virtual Tools game. More broadly, this model provides a mechanism for explaining how people condense general physical knowledge into actionable, task-specific plans to achieve flexible and efficient physical problem-solving.

4.1 Introduction

Imagine the following scenario: while trying to set up a tent on a camping trip, you realize that the ground is too hard for the tent stakes, and you have no hammer. What would you do? You might look around for a suitable hammer substitute, passing over objects like pinecones or water bottles in favor of a graspable rock. And if that rock failed to drive in the stakes at first, you might try a different grip, or search for a heavier rock. Most likely, you would only need a handful of attempts before you found an approach that works.

Determining how to pound in tent stakes without a hammer is an example of the flexibility and efficiency of more general physical problem solving. It requires a causal understanding of how the physics of the world works, and sophisticated abilities for inference and learning to construct plans that solve a novel problem. Consider how, when faced with the tent stake challenge, we do not choose an object at random; we choose a rock because we believe we know how we could use it to generate sufficient force on the stake. And if we find that the first rock fails, we again search around for a solution, but use the knowledge of our failures to guide our future search. This style of problem solving is a very structured sort of trial-and-error learning: our search has elements of randomness, but within a plausible solution space, such that the goal can often be reached very quickly.

Here we study the cognitive and computational underpinnings of flexible tool use. While human tool use relies on a number of cognitive systems – for instance, knowing how to grasp and manipulate an object, or understanding how a particular tool is typically used – here we focus on "mechanical reasoning," or the ability to spontaneously repurpose objects in our environment to accomplish a novel goal [166, 165, 84].

We target this mechanical reasoning because it is the type of tool use that is quintessentially human. While other animals can manipulate objects to achieve their aims, only a few species of birds and primates have been observed to spontaneously use objects in novel ways, and we often view these activities as some of the most "human-like" forms of animal cognition [197, e.g., Fig. 4-1A,B;]. Similarly, while AI systems have become increasingly adept at perceiving and manipulating objects, none perform the sort of rapid mechanical reasoning that people do. Some artificial agents learn to use tools from expert demonstra-

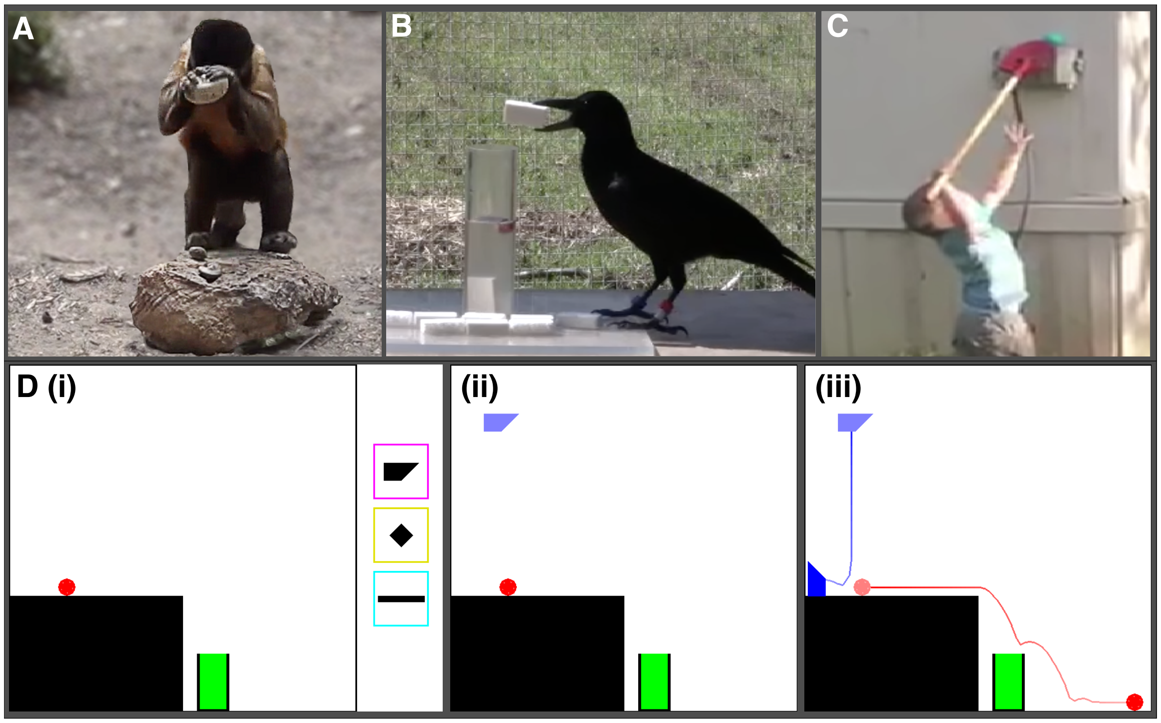


Figure 4-1: Examples of using objects to achieve a goal. **(A)** Bearded capuchin monkey opening a cashew nut with an appropriately sized stone [140]. **(B)** New Caledonian crow using heavy blocks to raise the water level in a tube in order to retrieve food [113]. **(C)** Toddler using a shovel to reach a ball (from youtu.be/hwrNQ93-568?t=198). **(D)** One illustrative trial in the Virtual Tools game (https://sites.google.com/view/virtualtoolsgame). (i) The player must get the red object into the green goal using one of the three tools. (ii) The player chooses a tool and where to place it. (iii) Physics is turned "on" and the tool interacts with other objects. The action results in a near miss.

tions [239], which limits their flexibility. Others learn from thousands of years of simulated experience [10], which is significantly longer than required for people. Still others can reason about mechanical functions of arbitrary objects but require perfect physical knowledge of the environment [218], which is unavailable in real-world scenarios. In contrast, even young humans are capable tool users: by the age of four they can quickly choose an appropriate object and determine how to use it to solve a novel task [17, e.g., picking a hooked rather than straight pipe cleaner to retreive an object from a narrow tube, Fig. 4-1C;].

What are the cognitive systems that let us use tools so flexibly, and accomplish our goals so rapidly? It has been suggested that mechanical reasoning relies on mental simulation, which lets us predict how our actions will cause changes in the world [166]. This general purpose simulation is a necessary component that supports our ability to reason

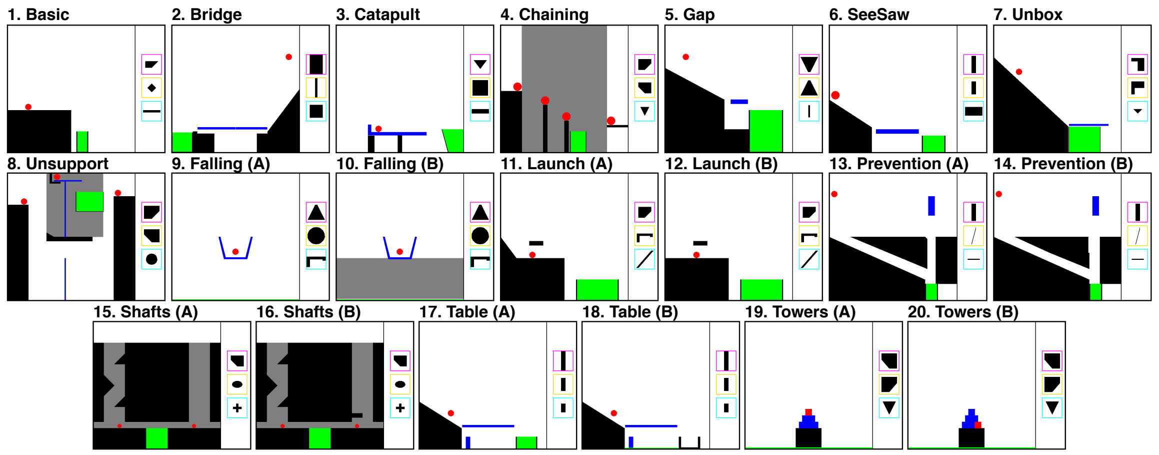


Figure 4-2: Twenty levels used in the Virtual Tools game. Players choose one of three tools (shown to the right of each level) to place in the scene in order to get a red object into the green goal area. Black objects are fixed, while blue objects also move; grey regions are prohibited for tool placement. Levels denoted with A/B labels are matched pairs.

about objects in novel environments, but by itself cannot explain how we make and update our plans so quickly. We propose that another key to rapid tool use is knowing what sorts of actions to even consider – both from an initial understanding of what actions are useful, and by updating this belief from observing the outcome of our actions, in simulation and in reality.

This paper makes two contributions. First, we introduce the Virtual Tools game, which presents a suite of physical problem solving challenges, and allows for precise, quantifiable comparisons between human and machine agents. Second, we present a minimal model of flexible tool use, called “Sample, Simulate, Update” (SSUP). This model is built around an efficient albeit noisy simulation engine that allows the model to act *flexibly* across a wide variety of physical tasks. To solve problems *rapidly*, the SSUP model contains rich knowledge about the world in the form of a structured prior on candidate tools and actions likely to solve the problem, which allows it to limit its simulations to promising candidates. It further learns from its simulations and from observing the outcome of its own actions to update its beliefs about what those promising candidates should be. Across 30 Virtual Tools levels in two experiments, we show that an instantiation of the SSUP model captures the relative difficulties of different levels for human players, the particular actions performed to attempt to solve each level, and how the solution rates for each level evolve.

4.1.1 The Virtual Tools game

Inspired by human tool use, as well as mobile physics games [1], we propose the Virtual Tools game as a platform for investigating the priors, representations, and algorithms used in physical problem solving (https://sites.google.com/view/virtualtoolsgame). This game asks players to place one of several objects ("tools") into a two-dimensional dynamic physical environment in order to achieve a goal: getting a red object into a green region (Fig. 4-1D). This goal is the same for every level, but what is required to achieve it varies greatly. Once a single tool is placed, the physics of the world is enabled so that players see the effect of the action they took. If the goal is not achieved, players can "reset" the world to its original state and try again; they are limited to a single action on each attempt. We designed 30 levels – 20 for the original experiment (Fig. 4-2) and 10 for a validation experiment (Fig. 4-7A) – to test concepts such as 'launching', 'blocking', and 'supporting'. Of the first 20 levels, 12 were constructed in six 'matched pairs' which incorporated small differences in the goals or objects in the scene to test whether subtle differences in stimuli would lead to observable differences in behavior.

The Virtual Tools game presents particular challenges that we believe underlie the kinds of reasoning required for rapid physical problem solving more generally. First, there is a **diversity of tasks** that require different strategies and physical concepts to solve, but employ shared physical dynamics that approximate the real world. Second, the game requires **long-horizon causal reasoning**. Since players can only interact with the game by placing a single object, they must be able to reason about the complex cause and effect relationships of their action long into the future when they can no longer intervene. Finally, the game elicits **rapid trial-and-error learning** in humans. Human players do not generally solve levels on their first attempt, but also generally do not require more than 5-10 attempts in order to succeed. People demonstrate a wide range of problem-solving behaviors, including "a-ha" insights where they suddenly discover the right idea for how to solve a particular task, as well as incremental trial-and-error strategy refinement. Figure 4-3 demonstrates how this occurs in practice, showing four different examples of participants learning rapidly or slowly, and discovering different ways to use the tools across a variety of levels.

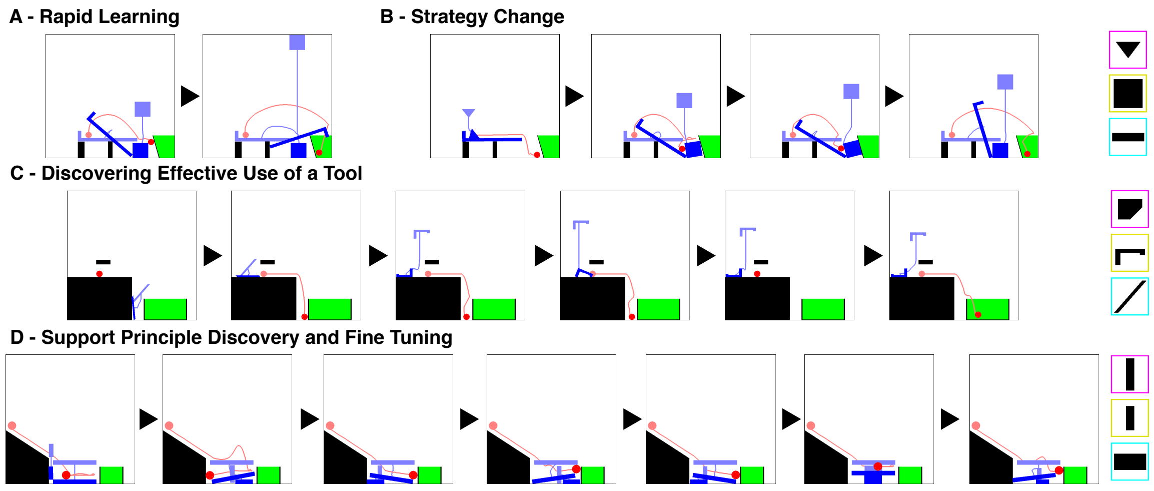

A - Rapid Learning
B - Strategy Change
C - Discovering Effective Use of a Tool
D - Support Principle Discovery and Fine Tuning

Figure 4-3: Examples of participants' behavior on three levels, representative of rapid trial-and-error learning: Initial plans are structured around objects, followed by exploring to identify more promising strategies and then refining actions until success. Objects start as shown by light blue/red outlines and follow paths traced out by colored lines. Possible tool choices shown to the right. **(A)** In the Catapult level, a useful strategy is often identified immediately and rapidly fine-tuned. **(B)** Other participants first try an unsuccessful strategy but then switch to a more viable strategy and refine it. **(C)** The Launch (B) level is designed to prevent obvious solutions. This participant may have initially believed the ball would start rolling and attempted to use a tool as a bridge. When this failed they realized they needed to launch the ball, but only discovered after several trials how to use a tool in a non-obvious way to accomplish this, via a hooking motion around the blocking ledge. They then took several more trials to fine-tune this action. **(D)** In the SeeSaw level, a participant realized on the second attempt they must support the platform for the ball to roll across, then tried different ways of making this happen.

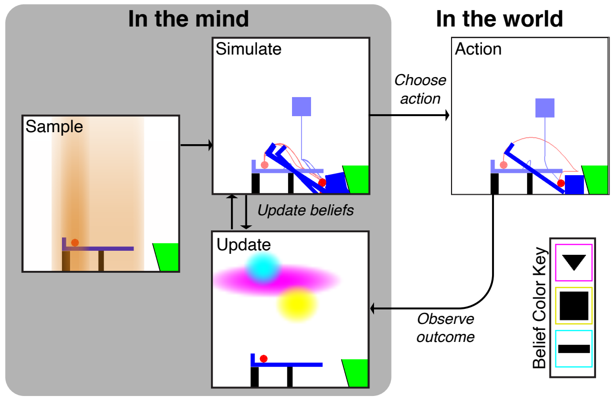
4.1.2 *Sample, Simulate, Update* Model (SSUP)

We consider the components required to capture both the flexibility and efficiency of human tool use. We propose that people achieve flexibility through an internal mental model that allows them to imagine the effects of actions they may have never tried before ("Simulate"). However, a mental model alone is not sufficient – there are far too many possible actions that could be simulated, many of which are uninformative and unlikely to achieve a specific goal. Some mechanism for guiding an internal search is necessary to focus on useful parts of the hypothesis space. We therefore propose people use structured, object-oriented priors ("Sample") and a rapid belief updating mechanism ("Update") to guide search towards promising hypotheses. We formalize human tool use with these components as the

A

```
Algorithm 1 SSUP algorithm
  Sample actions from prior $a \sim \pi(s)$
  Simulate action to get noisy rewards $\hat{r} \sim model(s, a)$
  Initialize distribution $\pi'(s)$ using samples $\hat{r}, a$
  while not successful do
    Sample action $a$
    Simulate action to estimate noisy reward $\hat{r} \sim model(s, a)$
    if $\hat{r} >$ threshold then
      Try action $a$ in environment
      Observe $r$
      If successful, exit
      Update policy with $r, s, a$.
    else
      Update policy with $\hat{r}, s, a$
    end if
  end while
```

B



C

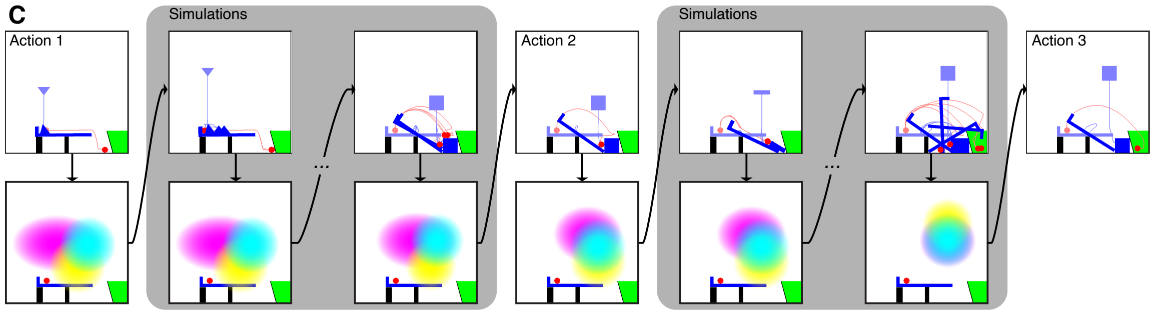


Figure 4-4: **(A)** The SSUP algorithm. **(B)** A diagram of the model for the Virtual Tools game. It incorporates an object-based prior, a simulation engine for filtering proposals, and an update module that suggests new proposals based on observations "in the mind" and from actions taken in the world. **(C)** Illustration of the policy π' evolving while attempting a level. Colored patches represent the Gaussian policy for each tool.

"Sample, Simulate, Update" model (SSUP; Fig. 4-4A).

SSUP is inspired by the theory of "problem solving as search" [161], as well as Dyna and other model-based policy optimization methods [211, 50]. Crucially, we posit that structured priors and physical simulators must already be in place in order to solve problems as rapidly as people; thus unlike most model-based policy optimization methods, we do not perform online updates of the dynamics model.

We want to emphasize that we view SSUP as a general modeling framework for physical problem solving, and only present here one instance of that framework: the minimal model (described below, with more detail in Appendix, Section A.2) that we think is needed to capture basic human behavior in the Virtual Tools game. In the discussion we highlight ways the model will need to be improved in future work, as well as aspects of physical reasoning that rely on a richer set of cognitive systems going beyond the framework presented here.

Sample: object-based prior At a minimum, the actions we should consider to achieve any goal should have the potential to impact our environment. We therefore incorporate an object-based prior for sampling actions. Specifically, the model selects one of the movable objects in the scene, then chooses an x-coordinate in an area that extends slightly beyond the width of the object, and a y-coordinate either above or below that object (Fig. 4-4B: Prior). For tool choice, we assume participants are equally likely to choose any of the three tools since all tools in the game were designed to be unfamiliar to participants. Samples from this distribution are used to initialize search.

Simulate: a noisy physics engine In order to determine which sampled actions are worth trying in the world, we assume people use an "Intuitive Physics Engine" [15] to flexibly imagine the effects of their actions. This engine is able to simulate the world forwards in time with approximately correct but stochastic dynamics [202, 183]. Determining the effect of a proposed action therefore involves applying that action to one's mental representation, and using the Intuitive Physics Engine to posit the range of ways that action might cause the world to unfold [37, 43]. Here we implement simulation using a game physics engine with noisy dynamics. People characteristically have noisy predictions of how collisions will resolve [202], and so for simplicity we assume uncertainty about outcomes is driven only by noise in those collisions (the direction and amount of force that is applied between two colliding objects).[1]

Since the internal model is imperfect, to evaluate an action we produce a small number of stochastic simulations (n_{sims}, set here at 4) to form a set of hypotheses about the outcome. To formalize how good an outcome is (the *reward* of a given action), we borrow an idea from the causal reasoning literature for how people conceptualize "almost" succeeding [78]. "Almost" succeeding is not a function of the absolute distance an action moved you towards your goal, but instead how much of a *difference* that action made. To capture this, the minimum distance between the green goal area and any of the red goal objects is recorded; these values are averaged across the simulations and normalized by the minimum distance that would have been achieved if no tool had been added. The reward used in SSUP

[1]We also considered models with additional sources of physics model uncertainty added, but found that the additional parameters did not improve model fit, so we do not analyze those models here.

is 1 minus the normalized distance, so that closer objects lead to higher reward.

Once the model finds a good enough action (formalized as the average reward being above some threshold), it takes that action "in the world." Additionally, to model time limits for thinking, if the model considers more than T different action proposals without acting (set here at 5), it takes the best action it has imagined so far. We evaluate the effect of all parameter choices in a sensitivity analysis (see Appendix, Fig. A-1).

Update: learning from thoughts and actions So far we have described a way of intelligently initializing search to avoid considering actions that will not be useful. But what if the prior still presents an intractably large space of possible actions?

To tackle this, we incorporate an update mechanism that learns from both simulated and real experience to guide future search towards more promising regions of the hypothesis space [115]. This is formally defined as a Gaussian mixture model policy over the three tools and their positions, $\pi'(s)$, which represents the model's belief about high value actions for each tool. $\pi'(s)$ is initialized with samples from the object-oriented prior, and updated using a simple policy gradient algorithm [231]. This algorithm will shape the posterior beliefs around areas to place each tool which are expected to move target objects close to the goal, and are therefore likely to contain a solution. Such an update strategy is useful when it finds high value actions that are nearby successful actions, but may also get stuck in local optima where a successful action does not exist. We therefore use a standard technique from reinforcement learning: epsilon-greedy exploration. With epsilon-greedy exploration, potential actions are sampled from the policy 1 - ε% of the time, and from the prior ε% of the time. Note that this exploration is only used for proposing internal simulations; model actions are chosen based on the set of prior simulation outcomes. This is akin to thinking of something new, instead of focusing on an existing strategy.

4.2 Results

We analyze human performance on the first 20 levels of the Virtual Tools game and compare humans to the SSUP model and alternates, including SSUP models with ablations and

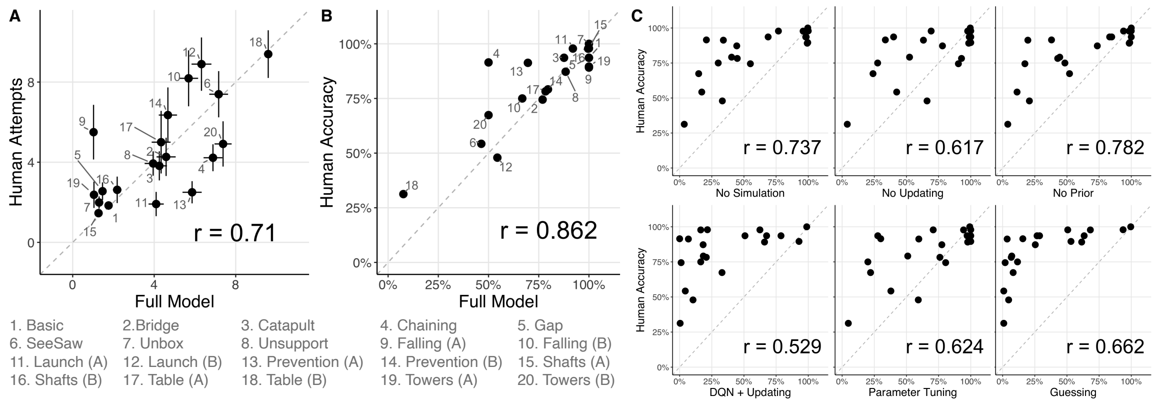


Figure 4-5: **(A)** Comparison of average number of human participants' attempts for each level with average number of attempts for the SSUP model. Bars indicate 95% confidence intervals on estimates of the means. **(B)** Comparison of human participants' accuracy on each trial versus the accuracy of the SSUP model. **(C)** Comparison of human participants' accuracy to all alternate models. Numbers correspond to the trials in Fig. 4-2.

two alternate learning baselines. We show that the full SSUP model best captures human performance. Access to the game and all data including human and model placements is provided at `https://sites.google.com/view/virtualtoolsgame`.

4.2.1 Human results

Experiments were approved by the MIT Committee on the Use of Humans as Experimental Subjects under protocol #0812003014. Participants were notified of their rights before the experiment, were free to terminate participation at any time by closing the browser window, and were compensated monetarily for their time.

We recruited 94 participants through Amazon Mechanical Turk and asked each participant to solve 14 levels: all 8 of the unmatched levels, and one variation of each of the 6 matched pairs (randomly selected).

Participants could choose to move on once a problem was solved, or after two minutes had passed. See Appendix, Section A.1 for further details.

The variation in difficulty between levels of the game was substantial. Participants showed an average solution rate of 81% (sd = 19%), with the range covering 31% for the hardest level to 100% for the easiest. Similarly, participants took an average of 4.5 actions (sd = 2.5) for each level, with a range from 1.5 to 9.4 average attempts. Even within trials, there was a large amount of heterogeneity in the number of actions participants used to solve the level. This would be expected with "rapid trial-and-error" learning: participants who initially tried a promising action would solve the puzzle quickly, while others explored different actions before happening on promising ones (e.g., Fig. 4-3).

Behavior differed across all six matched level pairs. We study whether these subtle differences do indeed affect behavior, even without feedback on the first action, by asking whether we can identify which level variant each action came from. We find these actions are differentiable across matched levels in 'Shafts', 'Prevention', 'Launch' and 'Table' on the first attempt, but not 'Falling' or 'Towers' (see Appendix, Fig. A-11 and Sec. A.6.1 for details). However, participants required a different number of actions to solve every level (all $ts > 2.7$, $ps < 0.01$). This suggests that people are paying attention to subtle differences in the scene or goal to choose their actions.

4.2.2 Model results

We investigate several metrics for comparing the models to human data. First, we look at how quickly and how often each model solves each level, and whether that matches participants. This is measured as the correlation and root mean squared error (RMSE) between the average number of participant attempts for each level and the average number of model attempts for each level, and the correlation and RMSE between human and model solution rates. The SSUP model explains the patterns of human behavior across the different levels well (Appendix, Table A.2). It uses a similar number of attempts on each level ($r = 0.71$; 95% CI $= [0.62, 0.76]$; mean empirical attempts across all levels: 4.48, mean model attempts: 4.24; Fig. 4-5A) and achieves similar accuracy ($r = 0.86$; 95% CI $= [0.76, 0.89]$; Fig. 4-5B).

Across many levels, the SSUP model not only achieves the same overall solution rate as people, but approaches it at the same rate. We measure this by looking at cumulative solution rates – over all participants or model runs, what proportion solved each level within X placements – and find that people and the model often demonstrate similar solution profiles (Fig. 4-6A; see Appendix, Section A.6.2 for quantitative comparison).

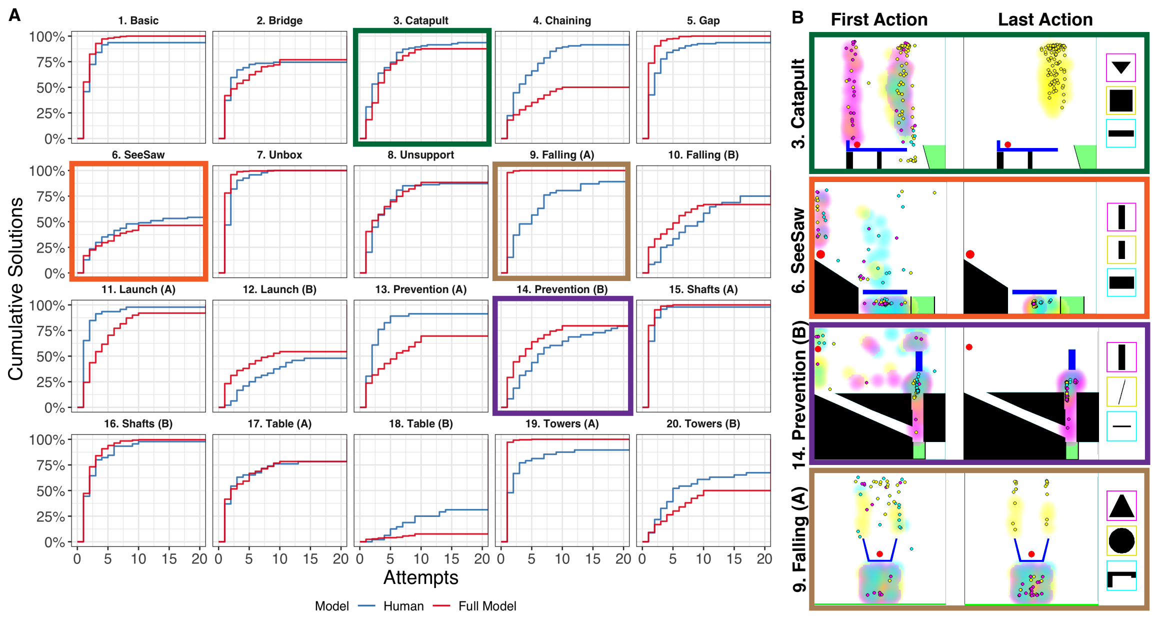


Figure 4-6: **(A)** Cumulative solution rate over number of placements for participants vs. the SSUP model. **(B)** Distribution of model actions (background) versus human actions (points) on the first and last attempts of the level for a selection of four levels. The distribution of model actions is estimated based on fitting a Kernel Density Estimate to the actions taken by the model across 250 simulations. Colors indicate the tool used, with the tools and associated colors shown to the right of each level. In most levels, the SSUP model captures the evolution of participants' solutions well, including the particular actions chosen; in the few cases that it differs, there is no alternative model that systematically explains these differences.

We can look in more detail how the model accomplishes this by comparing both the first actions that people and the model takes (Fig. 4-6B), and the actions that both take to solve a level (Fig. 4-6C). Like our human participants, the model takes significantly different actions on the first attempt between matched level pairs (see Appendix, Sec. A.6.1). More generally, both people and the model will often begin with a variety of plausible actions (e.g., Catapult). In some cases, both will attempt initial actions that have very little impact on the scene (e.g., SeeSaw and Prevention (B)); this could be because people cannot think of any useful actions and so decide to try *something*, similar to how the model can exceed its simulation threshold. However, in other cases, the model's initial predictions diverge from people, and this leads to a different pattern of search and solutions. For instance, in Falling (A), the model quickly finds that placing an object under the container will reliably tip the ball onto the ground, but people are biased to drop an object from above. Because of this, the model often rapidly solves the level with an object below, whereas a proportion of participants find a way to flip the container from above; this discrepancy can also be seen in the comparison of number of attempts before the solution, where the model finds a solution quickly, while people take a good deal longer (Fig. 4-5A). For comparisons of the first and last actions across all levels, see Appendix, Fig. A-11.

Model comparisons on Virtual Tools

We compare the full SSUP model against a set of six alternate models. Three models investigate the contribution of each SSUP component by removing the prior, simulation, or updating individually. Two models propose alternate solution methods: learning better world models rather than learning over actions (Parameter Tuning) or replacing the prior and simulator with a learned proposal mechanism (DQN + Updating). The Parameter Tuning alternate model uses inference to learn object densities, frictions and elasticities from observed trajectories. The learned proposal mechanism corresponds to a model-free deep reinforcement learning agent [156] which is trained on a set of 4500 randomly generated levels of the game (see Appendix, Sec. A.5), and then updated online for each of the 20 testing levels using the same mechanism as SSUP. This model has substantially more experience with the environment than other models, and serves as a test of whether model-free methods can make use of this experience to learn generalizable policies that can guide rapid learning. Finally, we compare to a "Guessing" baseline for performance if an agent were to simply place tools randomly. See Fig. 4-5C and Appendix, Table A.2 for these comparisons.

Eliminating any of the three SSUP components causes a significant decrease in performance (measured as deviation between empirical and model cumulative solution curves; all bootstrapped $ps < 0.0001$; see Appendix, Sec. A.6.2, Fig. A-6 for further detail). The reduced models typically require more attempts to solve levels because they are either searching in the wrong area of the action space (No Prior), attempting actions that have no chance of being successful (No Simulation), or do not guide search towards more promising areas (No Updating).

DQN + Updating performs worst of all plausible alternate models, using the most actions and solving levels at a rate barely over chance. Because this is equivalent to the No Simulation model with a different prior, its poor performance suggests that generalized action policies cannot easily be learned from repeatedly playing similar levels (see Appendix, Sec. A.5).

Because the Parameter Tuning model is equivalent to the No Updating model except that the properties of the dynamics model can be learned in Parameter Tuning, comparing those two models allows us to test whether we need to assume that people are learning the dynamics of the world in this game. The fact that both models perform roughly equivalently (see Fig. 4-5C) suggests that we do not need this assumption here.

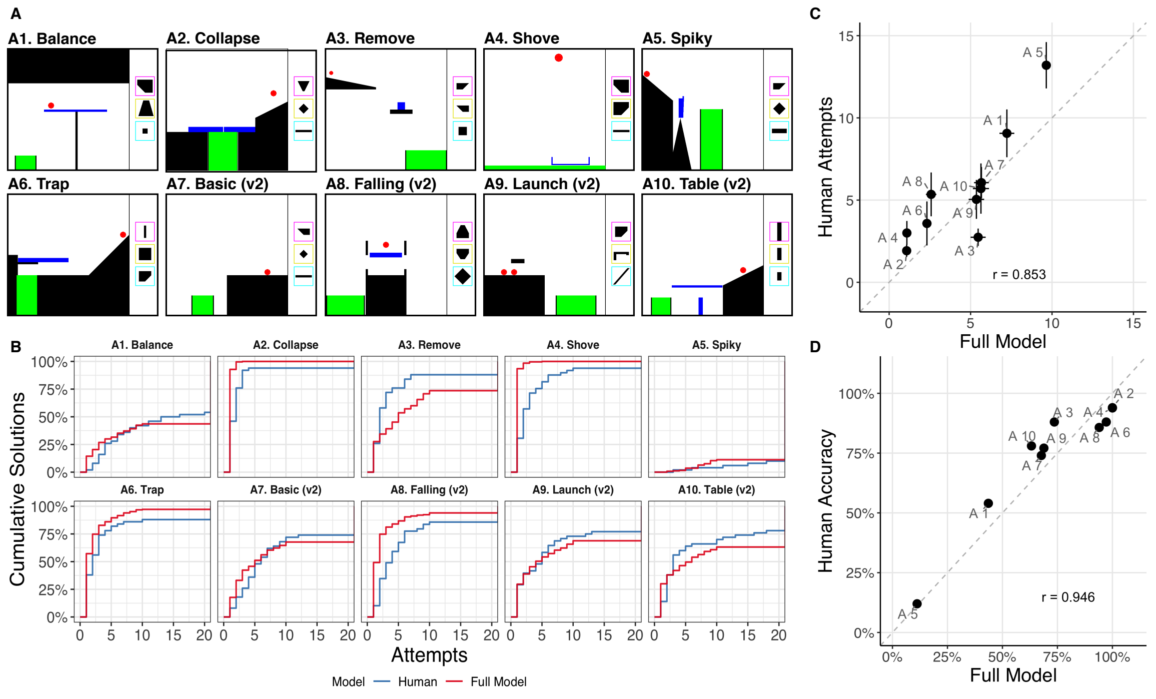


Figure 4-7: Results on 10 additional trials. **(A)** Trials used for the second experiment. **(B)** The cumulative solution rate for participants and the SSUP model. **(C)** Comparison of the number of human and model actions by trial. **(D)** Comparison of human and model accuracy on each trial.

Finally, we quantified how well each model captured the particular actions people took. Due to heterogeneity in participants' responses, we were unable to cleanly differentiate models' performance except to find that the DQN + Updating model underperformed the rest (see Appendix, Sec. S.6.3). However, no model reached the theoretical noise ceiling, suggesting components of the SSUP framework could be improved to better explain participants' actions (see the Discussion).

4.2.3 Validation on novel levels

We conducted a second experiment to test whether the models generalize to novel levels and physical concepts without tuning hyperparameters. For this experiment, we created 10 new levels: 6 novel level types and 4 variants of the originals (Fig 4-7A), testing an independent sample of 50 participants on all levels. The 6 novel level types were designed to test new physical strategies, including balancing, breaking, and removing objects from a ball's path. All other experimental details were identical to the main experiment.

Without tuning any model parameters, we find a good correspondence between human and model solution rates (Fig. 4-7B), and a strong correlation between the model's performance and human performance across number of placements (Fig. 4-7C, $r = 0.85$) and accuracy (Fig. 4-7D, $r = 0.95$). Similar to the main experiment, we find a decrement in performance if the prior or simulation are removed, or for the DQN + Updating model (all bootstrapped $ps < 0.0001$; Appendix, Fig. A-7). However, while numerically worse, we do not find a reliable difference if the update mechanism is removed ($p = 0.055$) or swapped for model learning ($p = 0.346$), suggesting that the particular reward function or update procedure might be less applicable to these levels (see Appendix, Sec. S.6.2).

4.3 Discussion

We introduce the Virtual Tools game for investigating flexible physical problem solving in humans and machines, and show that human behavior on this challenge expresses a wide variety of trial-and-error problem solving strategies. We also introduce a model for human physical problem solving: "Sample, Simulate, Update." The model presumes that to solve these physics problems, people rely on an internal model of how the world works. Learning in this game therefore involves condensing this vast world knowledge to rapidly learn how to act in each instance, using a structured trial-and-error search.

4.3.1 Model limitations

Although the SSUP model we used solves many of the levels of the Virtual Tools game in a human-like way, we believe that this is still only a first approximation to the rich set of cognitive processes that people bring to the task. In particular, there are at least two ways in which the model is insufficient: its reliance on very simple priors, and its planning and generalizing only in the forwards direction.

We can see the limits of the object-based prior in the Falling (A) level (Fig. 4-5B): people are much less likely to consider placing an object underneath the container to tip it over. Instead, many people try to tip it over from above, even though this is more difficult. In this way, people's priors over strategies are *context specific*, which causes them to be slower than the model in this level. In other cases, this context specificity is helpful: for instance, in the hypothetical level shown in Fig. 4-8A, there is a hole that one of the tools fits suspiciously perfectly into. Many people notice this coincidence quickly, but because the model cannot assess how tools might fit into the environment without running a simulation, it only succeeds 10% of the time. In future work, a more complex prior could be instantiated in the SSUP framework, but it remains an open question how people might form these context-specific priors, or how they might be shaped over time via experience.

People show greater flexibility than our model in the ability to work backwards from the goal state to find more easily solvable sub-goals [6]. In the hypothetical level in Fig. 4-8B, the catapult is finicky, which means that most catapulting actions will not make it over the barrier, and therefore will never hit the ball on the left. Instead, the easiest way to increase the objective function is by the incorrect strategy of knocking the ball on the right to get close to the goal, and therefore the model only solves the level 8% of the time. Working backwards to set the first sub-goal of launching the ball over the barrier would prevent getting stuck with knocking the ball as a local minimum. From an engineering standpoint, creating sub-goals is natural with discrete problem spaces [161], but it is less clear how these might be discovered in the continuous action space of the Virtual Tools game.

4.3.2 Related cognitive systems

There is an extensive body of research into the cognitive systems that underlie the use of real-world tools, including understanding how to manipulate them and knowing their typical uses [220, 165, 166, 17]. Here our focus was on “mechanical knowledge” of tools: how to use objects in novel situations. However, in real-world tool use, these systems work together with motor planning and semantic knowledge of tools. Future work can focus on these links, such as how novel tools become familiar, or how our motor limits constrain the plans we might consider.

The Virtual Tools game presents a problem solving task that blends facets of prior work, but encompasses a novel challenge. To rapidly solve these problems requires good prior knowledge of the dynamics – unlike Complex Problem Solving in which the dynamics are learned in an evolving situation [65] – and further iteration once a promising solution is considered – unlike the ‘a-ha’ moment that leads immediately to a solution in Insight Problem Solving [31, 82]. Unlike in traditional model-based or model-free reinforcement learning, in this task people bring rich models of the world that they can quickly tailor to specific, novel problems.

Distilling rich world knowledge to useful task knowledge is necessary for any agent interacting with a complex world. One proposal for how this occurs is “learning by thinking” [136]: translating knowledge from one source (internal models of physics) to another, more specific instantiation (a mapping between actions and outcomes on *this particular* level). We show how SSUP instantiates one example of “learning by thinking”: by training a policy with data from an internal model. Evidence for this sort of knowledge transfer has been found in people [74, 75], but has focused on simpler discrete settings in which the model and policy are jointly learned.

4.3.3 Virtual Tools as an AI Challenge

In preliminary experiments with model-free reinforcement learning approaches [156], we found limited generalization with inefficient learning across almost all of the Virtual Tools levels (see Appendix, Section A.5) despite significant experience with related levels.

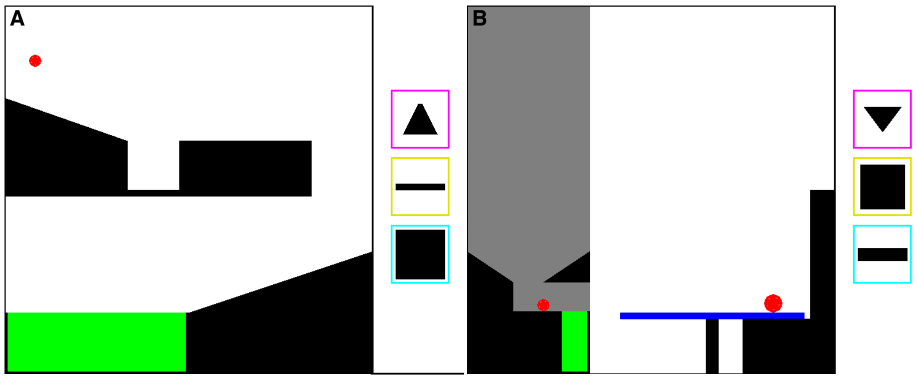


Figure 4-8: Two problems that demonstrate limitations of the current model. **(A)** A "suspicious conicidence" that one tool fits perfectly in the hole. **(B)** Creating a 'sub-goal' to launch the ball onto the other side is useful.

Based on our human experiments, we believe that model-based approaches will be required to be able to play games like Virtual Tools. Such approaches are becoming increasingly popular in machine learning [228], especially when combined with "learning-to-learn" techniques that can learn to adapt quickly to new tasks [55, 187]. Learning these models remains challenging, but approaches that incorporate added structure have excelled in recent years [28, 112]. Within the AI and robotics communities, model-based methods are already popular [218, 116, 68]. Remaining challenges include how to learn accurate enough models that can be used with raw sensor data [126], and how to handle dynamic environments.

Virtual Tools adds to a growing set of environments that test artificial agents' abilities to predict and reason using physics, such as the concurrently developed PHYRE benchmark [11] and others [229, 12, 71]. In contrast, our focus is on providing problems that people find challenging but intuitive, where solutions are non-obvious and do not rely on precise knowledge of world dynamics. By contributing human data to compare artificial and biological intelligence, we hope to provide a test-bed for more human-like artificial agents.

4.3.4 Future empirical directions

This work provides an initial foray into formalizing the computational and empirical underpinnings of flexible tool use, but there remains much to study. For instance, we do not find evidence that people learn more about the world, perhaps because here there is little benefit to additional precision here. But there are cases where learning the dynamics is clearly helpful (e.g., discovering that an object is abnormally heavy, or glued down), and we *would* expect people to update their physical beliefs in these cases. When and in what ways people update their internal models to support planning is an important area of study.

Children can discover how to use existing objects earlier than they can make novel tools [17], suggesting that tool creation is more challenging than tool use. Yet it is the ability to make and then pass on novel tools that is theorized to drive human culture [216]. It is therefore important to understand not just how people use tools, but also how they develop and transmit them, which we can study by expanding the action space of the Virtual Tools game.

4.3.5 Conclusion

Understanding how to flexibly use tools to accomplish our goals is a basic and central cognitive capability. In the Virtual Tools game, we find that people efficiently use tools to solve a wide variety of physical problems. We can explain this rapid trial-and-error learning with the three components of the SSUP framework: rich prior world knowledge, simulation of hypothetical actions, and the ability to learn from both simulations and observed actions. We hope this empirical domain and modeling framework can provide the foundations for future research on this quintessentially human trait: using, making, and reasoning about tools, and more generally shaping the physical world to our ends.

Chapter 5

Rapid Strategy Learning with Relational Program Policies

The previous chapter introduced the Virtual Tools game to study general physical problem solving in humans and machines, and showed that rapid trial-and-error learning in the game is made possible by a combination of mental simulation and structured action spaces. Within a particular problem, the "Sample, Simulate, Update" model accounts for both coarse- and fine-grained human behavior over the course of trying and failing to find a solution to a problem. However, neither humans nor the model learned *across* problems, which we attributed as weak evidence for the idea that people were learning how to use their simulators more effectively, rather than learning generically better world models to guide their actions. As alluded to in the discussion of the last chapter, here we investigate whether people might learn across levels if there are *strategies* that could be generalized between them; such as learning to "catapult" or "support" an object.

In a set of three different physical problem solving experiments, here we show that participants can learn adaptable, relational strategies that can be flexibly composed with novel physical variables even from very limited experience. The learned strategies are *contextual*, *physical* and *relational*. Participants selectively transfer these strategies when the crucial relational context of the strategy is satisfied, such as needing to "catapult", "support", "launch" or "destabilize" an object in the scene to accomplish a goal. We show that these strategies are composable with physical knowledge: people can adjust their strategies to account for new object weights despite no direct interaction experience with these objects. These empirical observations can be captured by a model that represents strategies as relational programs that guide the search procedure for a model-based planner, and learns these from purely trial-and-error experience. Taken together, these results suggest that people can make use of limited experience to learn abstract strategies that go beyond simple model-free policies and are instead object-oriented, adaptable, and can be parameterized by model-based variables such as weight.

5.1 Introduction

Imagine that you are attending a conference dinner buffet with a container of delicious-looking fried rice. As you go to scoop the rice onto your plate, you realize there is no serving spoon. The container is too hot to pick up, and being too polite to scoop the rice with your hands, what do you do? You might notice a stack of paper plates near the container that you can fold into a large scoop to pick up the rice. Having solved your rice problem, you could apply the same solution to other dishes without serving utensils. At future buffets with missing utensils, you would know to apply the same strategy, even if the plates were a different shape or made of a somewhat more rigid material, though you might have to adapt how you fold and scoop. And you might even transfer this strategy to scenarios that are nothing like buffets superficially, like using a soft Frisbee to dig a hole at the beach.

How are people able to learn and generalize these object oriented strategies so efficiently? Learning a simple, stereotyped scene-action response would not let you generalize this strategy beyond buffets with specific paper plates. Having general knowledge that paper plates are malleable does not tell you how you might use them to solve your problem. Instead it requires some insight about what the right kind of thing to do is, and how to adapt that idea to the new objects, scenes, and tasks you encounter. This applies not just to making novel paper utensils, but also to learning other kinds of abstract physical strategies such as levering heavy objects that could not otherwise be lifted, or placing an object under a wobbly table to stabilize it. Indeed, within tool use, children as young as four years of age are able to correctly adjust their strategy for interacting with a rake tool when the rake is no longer rigid [24], and switch to using a tool that obeys rigidity despite a lack of perceptual similarity.

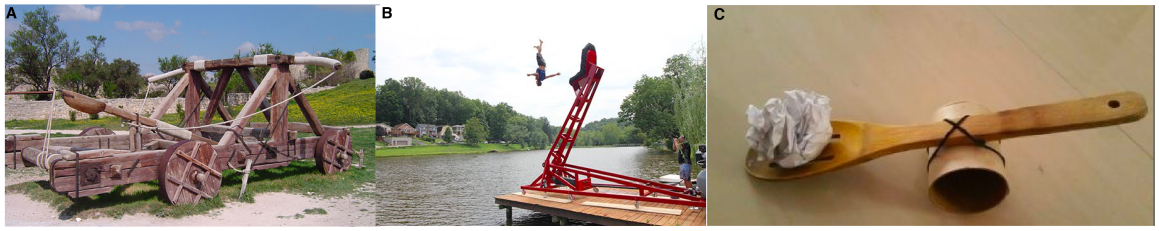


Figure 5-1: Examples of catapults across very different settings. **A** A medieval catapult (Figure credit: Vonmangle commonswiki at `https://commons.wikimedia.org/wiki/File:Replica_catapult.jpg`. **B** A human catapult. Figure credit: `verticalvisions.com`. **C** A catapult made from household materials. Figure credit: `https://gosciencegirls.com/diy-catapult-models/`.

Trial and error learning of this form is often modeled using either model-based or model-free reinforcement learning [74]. Model-based learning assumes that an agent is learning more about the dynamics of the world through its interactions, which allows it to form better plans in the future, e.g., learning more about the flexibility of the plates. But even with a perfect model of the world, there is a near-infinite set of actions one might take – how would you know that folding a plate is the correct thing to do, as opposed to searching elsewhere for utensils?

In contrast, one variant of model-free reinforcement learning assumes that learning guides an agent towards promising actions by updating a policy (which action or sequence of actions you should take from the current state to maximize reward). In this way the agent learns exactly what action to take in any given situation, and so does not have to consider the outcomes of large sets of actions. However traditional representations of policies are relatively brittle: common methods create look-up tables that represent one-to-one mappings of states and their features directly to values or actions [36, 74]. Such policy representations will not necessarily be flexible enough for an agent to transfer the paper plate skill to a new kind of paper, a new shape of plate, or possibly even a new buffet.

[5] found that people instead use a hybrid learning system for solving physical problems: they have pre-existing models of dynamics that they can use to assess the outcome of their actions [15], but learn what actions are useful in a given context by combining the

output of that model with observations from their own actions. However, while [5] found that people learn *within* a problem context, they found no evidence that people generalize *across* problems, perhaps because the physics puzzles they used in their study required a diverse set of strategies.

Here we ask whether people naturally transfer abstract policies (which we refer to as "strategies") for using objects across different contexts, and if so, what the representation of those strategies are. This is an important question because previous work focuses on how to reuse past experience by directly copying policies [36], or learning separate dynamics and reward functions from training tasks [64] by clustering *context* variables [237, 158]. A natural question is then: what are these context variables, and by which criteria should they be clustered together?

A possible answer comes from *relational* or *analogical* reasoning. The human capacity for analogy has been extensively studied in the context of inference: how do people use the structure of a problem to infer what knowledge can be transferred from a previous experience [73]? In these settings, the structure of the problem is given (by demonstration or verbal instruction) – participants must only determine a mapping that allows them to reuse knowledge from a previous structure.

This is not generally the task that people face. Instead, people must simultaneously discover for themselves both how to structure a problem in a way that *enables* transfer, as well as how to then map that structure onto new problems to most effectively reuse knowledge.

This is the setting in which our experiments lie. Through a set of three exploratory experiments, we examine what people learn when they are repeatedly exposed to similar physical problems, and what contexts they transfer that learning to. In Experiment 1, we show that people learn strategies, not generically better world models, that are selectively applied to problems that appear similar to the ones they encountered during training. In Experiment 2, we test the generalization and adaptability of these learned strategies by seeing whether people can apply them to scenes that look visually distinct from, but require the same physical concepts of, the training problems. Finally, in Experiment 3, we test whether learned strategies are model-based in that they can incorporate variables such as an object's weight. People trained on medium weight objects were able to immediately adapt their strategies to account for objects which were heavier or lighter than those experienced during training despite no direct interaction experience with the new objects. We demonstrate that these results can be explained by a computational model that learns *relational policies* represented as *logical programs* which provide an action prior for a model-based reinforcement learning agent. Computational models that do not represent policies relationally, or do not incorporate a noisy physical simulation engine, are unable to capture the same pattern of results we see in people. Together, these results suggest that people can simultaneously learn problem structure while also transferring it to new problems. This structure is inherently relational, and people can adapt strategies to new, learned physical model variables to maintain flexibility.

5.1.1 Relational Program Policies

How do people learn useful policies across tasks, while still maintaining flexibility within tasks? Models from analogical reasoning suggest people map *structure* across tasks to find similarities which allow them to make inferences [73, 151]. On the other hand, models from reinforcement learning suggest people learn policies that can generalize across tasks over time [217], but these are unstructured, and so may be fairly limited in their generalization. To explain how people can simultaneously learn policies within tasks while also learning structure across tasks, we must integrate these two perspectives.

Our method takes inspiration from *relational reinforcement learning* within Artificial Intelligence (relational RL [53]) to integrate these two perspectives on learning and transfer. In relational reinforcement learning, states and actions are represented in a *relative* way. For example, imagine a tower of blocks. In traditional setups, you might represent the tower by listing the position of each block in absolute coordinates. In relational RL, you would instead represent these states by listing the relations that hold between blocks, such as one block being on top of another (*above(block1, block2)*). This can lead to significantly improved transfer across scenes. If the block tower is simply moved to another table, all of the absolute positions of the blocks might change, but the relations between them will remain constant. If you represent the problem relationally, this new scene will then appear identical to the problem you just solved, meaning that you could simply *reuse* your previous solution instead of solving the problem again from scratch.

Despite the promise of relational RL, there are at least two limitations that have prevented its more widespread adoption. First, relational reinforcement learning is often prone to over-fitting. One of the most popular methods, TILDE-RT [53] uses decision trees in conjunction with Q learning to learn a single representation of the Q value for (state,action) pairs as a decision tree. This can lead to catastrophic overfitting. If the reward landscape in state or action space is not smooth, the decision tree will quickly propose a much too narrow region of space as the area of highest reward. Since many problems do not have such smooth reward landscapes, this is a substantial limitation. Second, the number of relational predicates that are used in relational RL is *fixed* and *finite*. To truly capture human learning, we might want our methods to be amenable to potentially infinite numbers of relational predicates, as well as generalizing to potentially infinite action spaces.

To overcome these limitations, we present *Relational Program Policies* (RPP), which combines the best of relational reasoning models from analogy with the structure learning capabilities of reinforcement learning. RPP represents policies as logical disjunctions of conjunctions over relational predicates. Unlike traditional relational RL, RPP can be interpreted as a form of Bayesian program induction. By approximating policies using a weighted Bayesian ensemble, RPP is less susceptible to the problems of over-fitting and sensitivity to noise in previous methods from relational RL. Because RPP is a form of program induction, it also has the potential to operate over *infinite spaces* of relational predicates.

5.1.2 Representations of policies in RPP

We follow [199] in defining a policy class Π consisting of state (S)-action(A) classifiers $h : \mathcal{S} \times \mathcal{A} \to \{0,1\}$. When $h(s,a) = 1$, the action a in state s is permissible and may be taken. By using this parameterization, h is a function that can be applied to sets of states and actions of any size – we simply run the function over all available $(state, action)$ pairs.

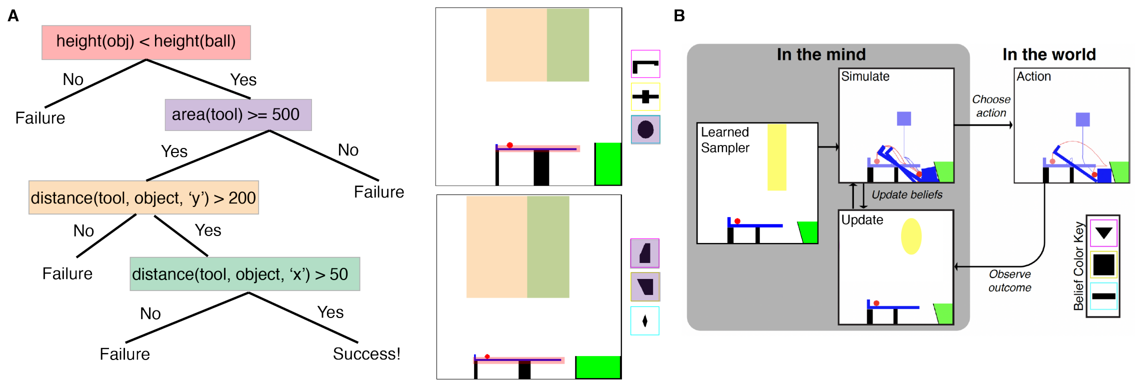


Figure 5-2: Relational Program Policies. **A** An example of a learned relational decision tree program for describing the strategy of catapulting an object into a goal. The colors in the tasks show how each relational predicate specifies a narrower region of the possible action space. **B** The relational program is incorporated into a *within-task* search procedure which samples actions specified by the relational program, imagines how they might impact the world, and then updates its estimates of which actions will be effective.

A policy $\pi(a \mid s)$ will then sample action a uniformly at random from the permissible set. If no action is permissible, the policy will sample an action at random.

How should h be represented? Relational reinforcement learning would write down a set of finite rules for h, and simply calculate whether or not each of those rules holds for the current scene. This is unnecessarily burdensome and rigid. Instead, we propose that the rules are *programmatic*: they are expressions in some domain specific language (DSL) that take states and actions as inputs. Writing down programmatic rules has two advantages. First, this allows the rule space to be potentially *infinite* – if rules are drawn from a grammar, that grammar could potentially ground out in infinite combinations, and it would not be a problem. Second, we do not have to make as many assumptions about the specific rules that people might be bringing to the task – these can be *discovered* as particular programs within the grammar.

However, there is a catch. There are now many more possible rules that could describe a good policy, so naive enumeration through the space of rule combinations would be infeasible. As in [199], to tackle this inference problem we restrict ourselves to only consider *logical combinations* of programmatic rules $f_i(s,a)$ drawn from the grammar, specifically "or"s of "and"s:

$$h(s,a) = (f_0(s,a) \wedge \ldots \wedge f_1(s,a)) \vee \ldots \vee (f_{n-1}(s,a) \wedge \ldots \wedge f_n(s,a)) \tag{5.1}$$

By making this restriction, we can take advantage of the same greedy learning algorithms that are used by traditional relational reinforcement learning, in this case decision trees [52].

5.1.3 Inference in RPP

RPP can be seen as performing Bayesian inference over the set of $(state, action)$ pairs that will lead to success given a set of examples $\mathcal{D}$ which are either failures or successes. With this perspective in mind, we can compute a full posterior distribution over policies $\mathsf{p}(\pi \mid \mathcal{D})$.

Probabilistic Model $\mathsf{p}(\pi, \mathcal{D})$

To define a joint distribution over policies π and data $\mathcal{D}$, we must define a prior $p(\pi)$ and likelihood $p(\mathcal{D} \| \pi)$. As in previous work, the prior is chosen to encode a preference for policies which use fewer and simpler programmatic rules [199]. Therefore the joint probability of a relational policy is simply the product of the probabilities of each of its programmatic rules $f_i(s,a)$ [144]. The probabilities for each $f_i(s,a)$ are simply the probability of generating that particular programmatic rule given the generation probabilities in the associated probabilistic context-free grammar $p(f)$ referred to in the last section and shown in Table 5.1.

The likelihood of a dataset $\mathcal{D}$ given a policy π is $p(\mathcal{D} \mid \pi) \propto \prod_{i=1}^{N} \prod_{j=1}^{T_i} \pi(a_{ij} \mid s_{ij})$.

Approximating the Posterior $\mathsf{p}(\pi \mid \mathcal{D})$

To calculate an approximate posterior $q(\pi) \approx p(\pi \mid \mathcal{D})$, we assume $q(\pi)$ to be a weighted mixture of K policies (μ_1, ..., μ_K) and initialize each μ_i to have equal weight and be equal to the uniform policy over actions. Assuming we have at least one positive example in $\mathcal{D}$, we can generate negative examples by considering counterfactuals over the action or state space. For simplicity, here we consider only counterfactuals of actions that choose a different object in the scene, or different geometric relationships with the object that was successfully acted upon. Given that we now have at least one positive and one negative example, we can then set up a classification problem to attempt to find programmatic rules $f_i(s,a)$ which will successfully distinguish the positive and negative examples.

Since the number of possible rules $f_i(s,a)$ is theoretically infinite, we enumerate through rules in order of probability. The rules with highest probability under the grammar are selected first, followed by increasingly more complex (and less probable) rules. This is straightforward to do with best first search. With this finite set of rules N, each $(state, action)$ pair in the dataset $\mathcal{D}$ can be converted into a length N vector with each entry representing the output of a particular rule. It is then straightforward to apply any classification algorithm to the dataset consisting of length N vectors paired with a success label [155, 177]. Here we follow [52] and use a decision-tree learner [172].

The inferred decision tree represents a candidate policy μ_* (see Figure 5-2A for an example). We can calculate the prior probability of μ_* based on which rules it chose to use, and evaluate the likelihood based on how well it classifies the available data. μ_* will be included in the posterior mixture q if its unnormalized posterior probability is greater than the lowest-scoring component in the current mixture. The mixture is then reweighted according to $q(\mu_i) = \frac{p(\mu_i|\mathcal{D})}{\sum_{j=1}^{K} p(\mu_j|\mathcal{D})}$. We stop iterating after a fixed number of steps, although other stopping criteria are also possible.

The optimal policy is then $\pi_*(s) = argmax_{a\in\mathcal{A}} \sum_{\mu\in \mathsf{q}} \mathsf{q}(\mu)\mu(a|s)$.

5.1.4 Using RPP to guide model-based search

Given a dataset of $(state, action)$ pairs, RPP can efficiently find a relational program that will differentiate between successful and unsuccessful actions. This can explain how people learn relational structure *across tasks* and use it to transfer strategies across disparate contexts. However, there is still a learning problem *within* tasks – how should I use my failures in this particular task to guide what I do next?

In previous work, [5] suggested that people use a framework called "Sample, Simulate, Update" (SSUP) which suggests that within tasks, people use a combination of noisy physical simulations and real actions to hone in on promising regions of action space. Critical to their model was a "sample" step which was assumed to be a generic action prior that preferentially picked actions which would interact with objects in the scene. We propose that to capture learning *across* tasks, it is sufficient to replace the "sampler" from their original framework with the learned relational program policy (Figure 5-2B). The data given to RPP is then all the simulations and actions that were attempted across previous levels where the agent was successful.

Combining RPPs with this model-based search procedure provides several advantages over the model-free RPP method from the previous section. RPPs can now adapt to changes to the physical world. If it turns out that the learned strategy is not applicable in this particular scene, the noisy simulator can guide the agent away from those regions proposed by RPP. Additionally, if something about the physical world changes which does not impact the relations between objects (for example, maybe you encounter a new object with greater mass, or a particularly slippery object), RPP + SSUP can immediately adapt, while RPP alone would take exactly the same action that worked for the old physical variables. For experiments in this chapter, we do not modify any of the SSUP hyperparameters – all hyperparameters are identical to those used in [5].

5.1.5 Alternate models

We compare two alternate models to RPP+SSUP to test both the importance of representing policies with *relational programs*, and the importance of using RPPs to guide *mental simulation* as opposed to using them in a model-free way.

"Non-relational model-based strategies" uses a notion of object-oriented actions, like the "Sample" procedure in the original SSUP model, but does not incorporate relational information into the policy. Mathematically, this means that the "Sample" procedure is represented as first choosing a dynamic object in the scene ($obj\ Categorical(\{dynamicobjects\})$), then choosing an offset to that object as a Gaussian centered upon it ($x, y\ location(obj) + Gaussian(0, \sigma_{obj})$). The tool is similarly sampled as a categorical distribution across options ($tool\ Categorical(tool_1, tool_2, tool_3)$. The parameters for each distribution are learned within a level, and transferred across levels based on object similarity (defined as $\frac{min(area(obj1),area(obj2))}{area(ConvexHull(obj1+obj2))}$). If an object in the new scene is sufficiently similar to one in the old scene, we transfer the Gaussian distribution parameters, σ_{obj}, as well as the probability of selecting that object under the Categorical distribution. The probabilities across categories are then re-normalized.

"Relational model-free strategies" ablates the model-based search procedure, but maintains the representation of policies as relational programs, and uses the same inference procedure as described in the preceding section. If the policy does not have at least one success and one failure to learn a program, it samples an action randomly from the same object-oriented "Sample" distribution as used in SSUP [5].

5.2 Results

In the following experiments, we highlight three aspects of RPP+SSUP. In Experiment 1, we show that people can learn relational strategies where the available tools and precise positions of objects in the scene change across tasks, and that these are selectively applied when the relational *context* of the scene is maintained. In Experiment 2, we introduce more visually distinct test tasks that still share the underlying relational context with the training tasks, demonstrating again that RPP+SSUP transfers strategies when appropriate, and learns efficiently in training. We additionally show that simpler alternatives to RPP+SSUP, such as ignoring relations and focusing solely on object similarity, cannot account for the learning curves or differences in test performance which people demonstrate. In Experiment 3, we test whether learned strategies are model-based in that they can incorporate variables such as an object's weight. People trained on medium weight objects were able to immediately adapt their strategies to account for objects which were heavier or lighter than those experienced during training despite no direct interaction experience with the new objects.

We focus on the Virtual Tools game proposed by [5], shown in Figure 5-1. Each problem is presented as an initial scene (with physics turned off), a goal description, and three 'tool' objects to choose from (Fig. 5-1A). Participants must accomplish the stated goal ob-

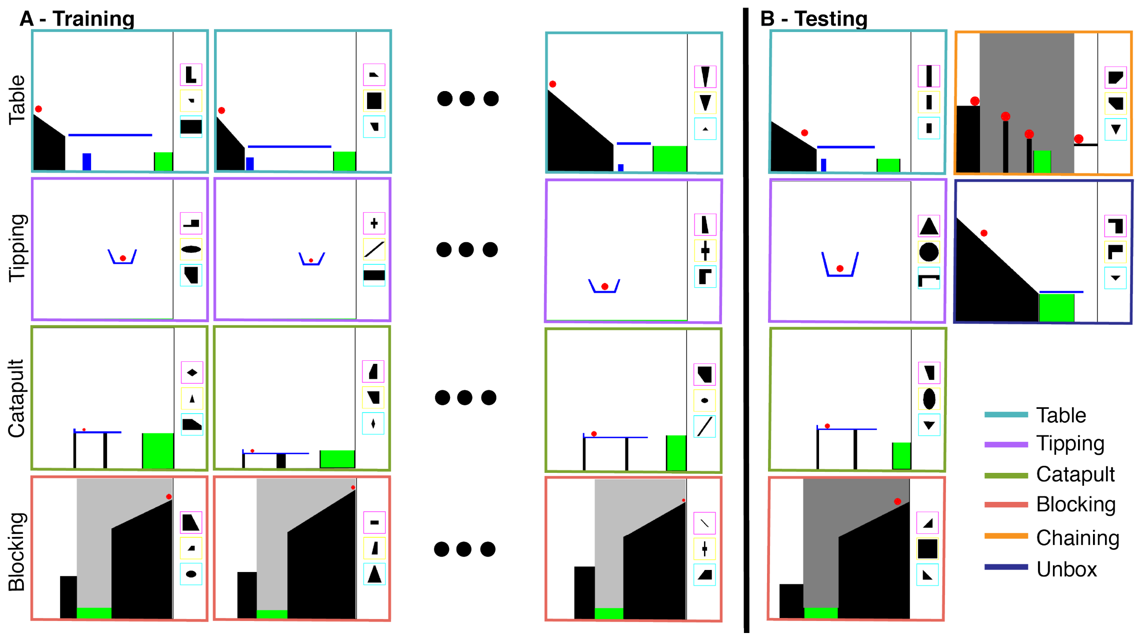


Figure 5-3: Diagram of Experiment 1. (A) Participants start with a learning phase where they play 10 related levels with random tool options with only three attempts per level. (B) Participants play the same set of 6 levels in random order during the testing phase. See `bit.ly/toolgame` to play the test levels without limits on the number of attempts.

jective by selecting one of the 'tool' objects and clicking to place it somewhere in the scene (Fig. 5-1B). After a single 'tool' is placed, physics is turned on using the Chipmunk 2D physics engine, and participants can see the resulting trajectories of all objects in the scene (Fig. 5-1C). Participants were given three attempts for each problem to accomplish the objective. We recorded all attempted actions including which tool was used and where it was placed.

We followed a similar experimental protocol to [5] in order to familiarize participants with the task. First, participants were given a set of initial instructions explaining the kinds of objects that might exist in the different problems. They then interacted with a 'playground' level without a goal, and were required to make 3 placements before moving on. Finally, participants were given two simple practice levels that they had to solve before moving on; these were not analyzed. Participants then continued on to the main body of the experiment described in the sections below.

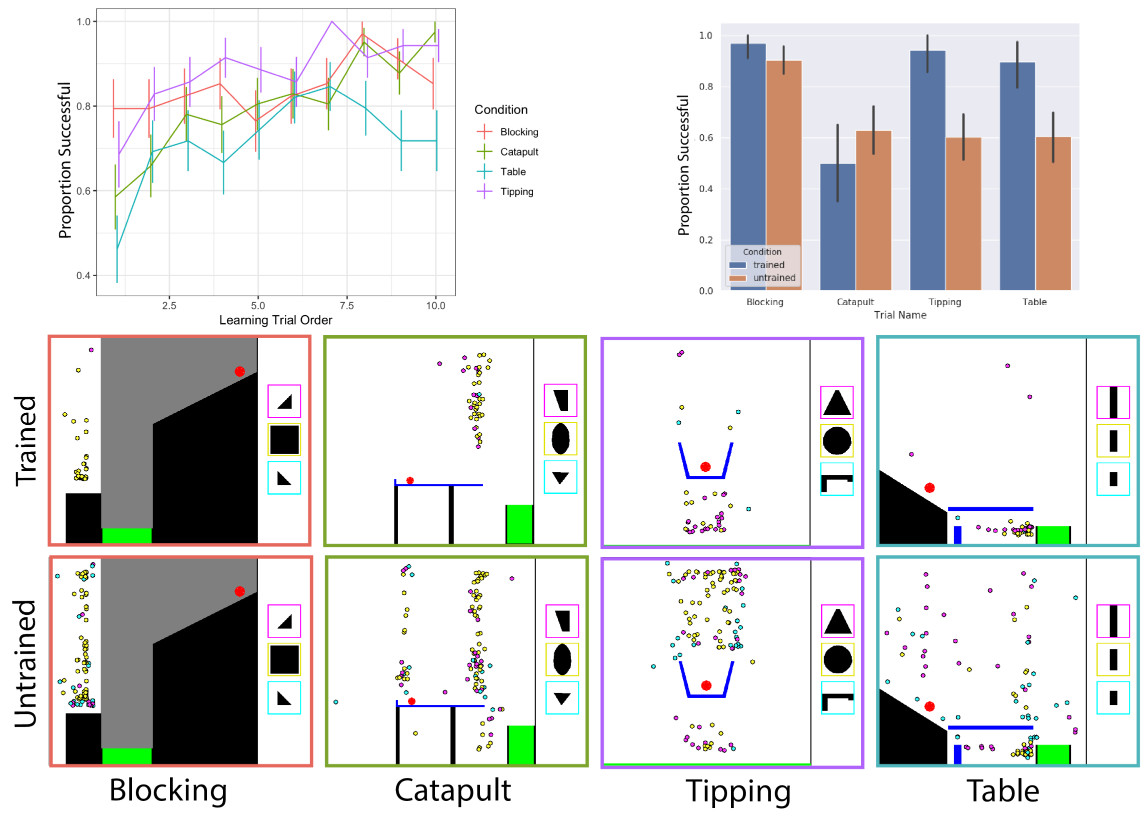


Figure 5-4: Results of Experiment 1. (Top left): Learning curves across the course of training, as measured by whether participants were successful within 3 attempts for a given trial. (Top right): Testing performance after training. Participants who were trained in the matched condition tend to outperform participants trained in other conditions, with the exception of “Catapult”. (Bottom): First placements for the test levels, split by participants trained on matching levels, and those trained in other conditions.

5.2.1 Experiment 1: Learning strategies

[5] found no evidence of learning when people played 14 unrelated levels of the Virtual Tools game. They proposed that people likely did not learn better models of the world while they interacted with it, but instead learned a policy which allowed them to make better use of a model in individual trials. Since the trials differed substantially across training, reusing these policies in new scenes would be mal-adaptive. Instead, it could be that participants will only reuse learned policies when those policies would be useful in accomplishing their goals.

Procedure

To test whether people could learn strategies, we designed four types of levels: *Catapulting*, *Tipping*, *Blocking* and *Tabling*. We then created 10 random variations of each level type where different aspects of the scene were varied, including the sizes and positions of each object and the shapes of each tool (see Fig 5-3A). Each participant was assigned to one of the four level types, and played all 10 random levels of that type. Participants were only given three attempts to solve each trial; if they did not succeed within those attempts, the level was marked as unsolved and they moved on to the next training level. The trial order was randomized across participants. After the training phase, all participants were given the same six testing problems: one from each of the four basic categories, and two chosen as controls to ensure that certain training conditions did not allow participants to become generically better at the game (Fig 5-3B). The two control levels were taken from [5]: *Chaining* was relatively difficult for participants, while *Unbox* was relatively easy. Just as in the training levels, participants were only allowed three attempts to solve the test levels.

153 participants were recruited using Amazon Mechanical Turk. We only analyze data from participants who succeeded on at least one of their training levels – these criteria excluded four participants.

Results

We first test whether there is any learning during training, and find that accuracy (whether a participant succeeded within 3 attempts) does improve over this phase (for all conditions, all $\chi^2(1) > 4.4$, all *ps* < 0.036; Fig. 5-4), thus suggesting that training was effective.

During testing, we find no evidence of difference in performance on the control trials based on training condition (*Unboxing*: $\chi^2(3) = 2.29,\ p = 0.51$, *Chaining*: $\chi^2(3) = 6.16,\ p = 0.10$ respectively). This suggests that participants were not learning more about the game in general, similar to how [5] found no transfer learning across different types of levels.

Instead, people behave differently in test levels depending on the level type they were trained on (Figure 5-4). Participants were reliably more accurate in the *Tipping* (94%

trained vs. 61% untrained; $\chi^2(1) = 17.5,\ p < 0.001$, odds-ratio = 10.8) and *Table* conditions, (90% trained vs. 61% untrained; $\chi^2(1) = 13.0,\ p < 0.001$, odds-ratio = 5.7), but not in the *Catapult* (50% trained vs. 63% untrained; $\chi^2(1) = 2.01,\ p = 0.16$, odds-ratio = 0.59) and *Blocking* (97% trained vs. 90% untrained; $\chi^2(1) = 1.93,\ p = 0.17$, odds-ratio = 3.52) conditions (Fig. 5-4B). While the *Blocking* condition can be explained by ceiling effects (all participants find this level easy), *Catapult* requires a more detailed analysis.

To understand why trained participants are not more accurate on the *Catapult* testing level, we examine the detailed placements people tried (Fig. 5-4). For this level there are 2 types of solutions: one solution type involves catapulting the ball into the goal, while the other involves hitting the ball directly and launching it into the goal. Participants in the *Catapult* training condition always try to use the catapulting strategy. However, in this particular level, that strategy is more difficult than directly hitting the ball. Participants who were not trained on *Catapult* hit the ball directly more often, and therefore had a slight advantage, leading to similar overall performance in terms of accuracy but easily distinguishable behavior.

What is learned? While we suggest that people are learning flexible strategies over objects and scenes, an alternate explanation could be that people are instead learning simple associations of where to place objects. This association must be more complex than simply placing objects in particular spatial locations, as the particular places that lead to success vary across each level. Instead, to improve in performance, they would need to learn an object-oriented strategy such as putting an object beneath the platform in *Table*, or above the lever in *Catapult*.

However, this does not rule out learning a simple object-oriented spatial prior on actions to take – e.g., placing the tool under or over the key object. We can examine whether a simple over/under prior is being learned by looking at how participants perform on *Tipping* – a test level that could be solved by placing the tool over or under an object – depending on whether they were trained on levels that require dropping a tool from above (*Catapult*) or placing a tool underneath an object (*Table*). If this simple strategy is learned, we would expect participants in the *Catapult* training condition to be more likely to place a tool above, whereas participants trained on *Table* levels would be more likely to place a tool below. However, we find no evidence for any differences: the same proportion of participants placed the tool above regardless of whether they were trained on *Catapult* (82%) or *Table* (80%; $\chi^2(1) = 0,\ p = 1$; Fig. 5-4D). Thus we find evidence that the strategies participants are learning are context specific.

5.2.2 Experiment 2: Generalizing strategies

How context specific are the learned strategies? In Experiment 2 we tested whether the context depends on the presence of *relational* concepts even when the problem appears visually distinct. Participants were randomly assigned to three training conditions (*Catapult*, *Tipping* and *Table*) and trained using the same 10 levels from Experiment 1. Each participant then played 6 testing levels: 3 matched testing levels from Experiment 1, and 3 "transfer" levels designed for each level type (Fig 5-5C).

The transfer levels were designed to maintain the same relational structure as the training levels but appear visually distinct. For example, in the *Catapult Transfer* level, one must drop a tool in order to catapult the ball so that it hits a smaller ball which will then roll into the goal, instead of catapulting an object into the goal directly. In the *Table Transfer* level, succeeding requires supporting two planks simultaneously by putting a large block underneath them. In the *Tipping Transfer* level, participants need to place an object underneath the platform to destabilize it, thus tipping the ball into the container. If trained participants solve their transfer level more efficiently, this suggests that the representation of strategies and when to apply them is based on *relational concepts* rather than simple visual similarity.

Sixty-two participants were recruited using Amazon Mechanical Turk. We only analyze data from participants who succeeded on at least one of their training levels – this caused four participants to be filtered from our analysis.

Results

We broadly replicate the results of Experiment 1, first finding a significant improvement in accuracy over the course of training (for all conditions, all $\chi^2(1) > 11.9$, all $ps < 0.001$). The important test for Experiment 2 involves looking at the test accuracy in the *generalization* conditions (Figure 5-5.). Here, in both the *Catapult Transfer* and *Table Transfer* levels, we see significantly better accuracy for participants trained on that level type (*Catapult Transfer*: 76% trained vs. 32% untrained; $\chi^2 = 9.58$, $p = 0.002$, odds-ratio = 6.75, *Table Transfer*: 100% trained vs. 63% untrained; $\chi^2 = 10.9$, $p < 0.001$). We do not find evidence of improvement for the *Tipping Transfer* level (39% trained vs. 52% untrained; odds-ratio = 0.59). We can again look at the fine-grained participant behavior to understand why this is.

During training on the *Tipping* levels, participants could solve each level using two distinct strategies. One strategy involves tipping the container from *above* with a precise placement on the side of the container to flip it over. The other strategy involves tipping the container from *below* by putting any object beneath the container to destabilize it. In Experiment 1, almost all participants found and used the "tipping from below" strategy since it is more robust. We expected a similar effect in Experiment 2, and so the *Tipping Transfer* level is *only* solvable by destabilizing the platform from below. However, in Experiment 2 a significant proportion of our participants found and consistently used the "tipping from above" strategy: 55% of participants in the *Tipping* condition never tried tipping the container from below. Instead, they became adept at tipping from above.[1] In Figure 5-7, we split participants by those who "tip from below" and those who "tip from above", and compared them to participants who were not trained in this condition. It is clear that (a) participants who learned to tip from above develop an exceptionally precise strategy compared to those who were untrained in this condition, and (b) if participants discover "tipping from below" they use it consistently and can generalize it to the transfer level reasonably well (with a mean accuracy of 70% compared to 0% for those "tipping from above" and 50% for those who were untrained).

[1]This could be due to the difference in the number of participants recruited: Experiment 1 had 40 participants in the *Tipping* condition while Experiment 2 only had 19.

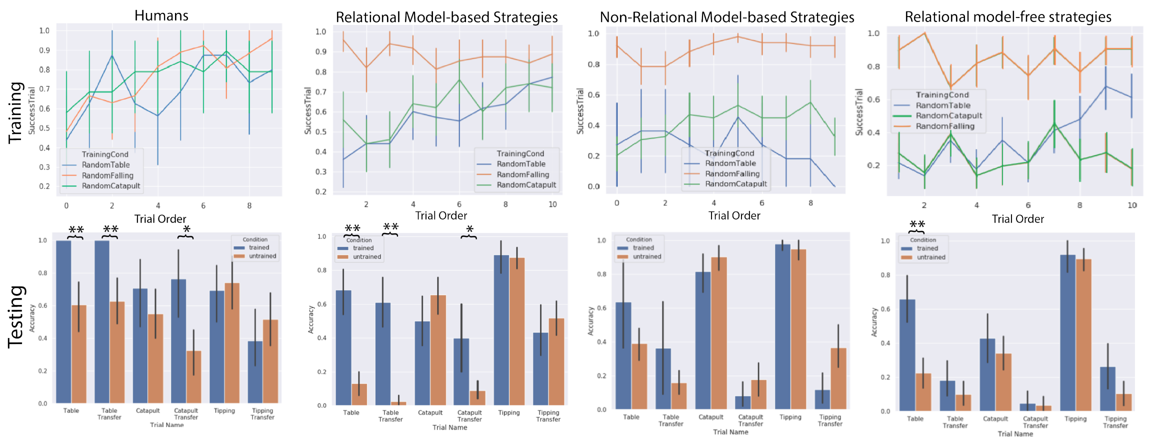


Figure 5-5: Performance in Experiment 2. (A) Proportion of successful participants on testing levels is generally higher for those trained on the strategy. (B) Specific placements and tool used on the first attempt for the transfer conditions in each strategy. People were trained on levels from Experiment 1.

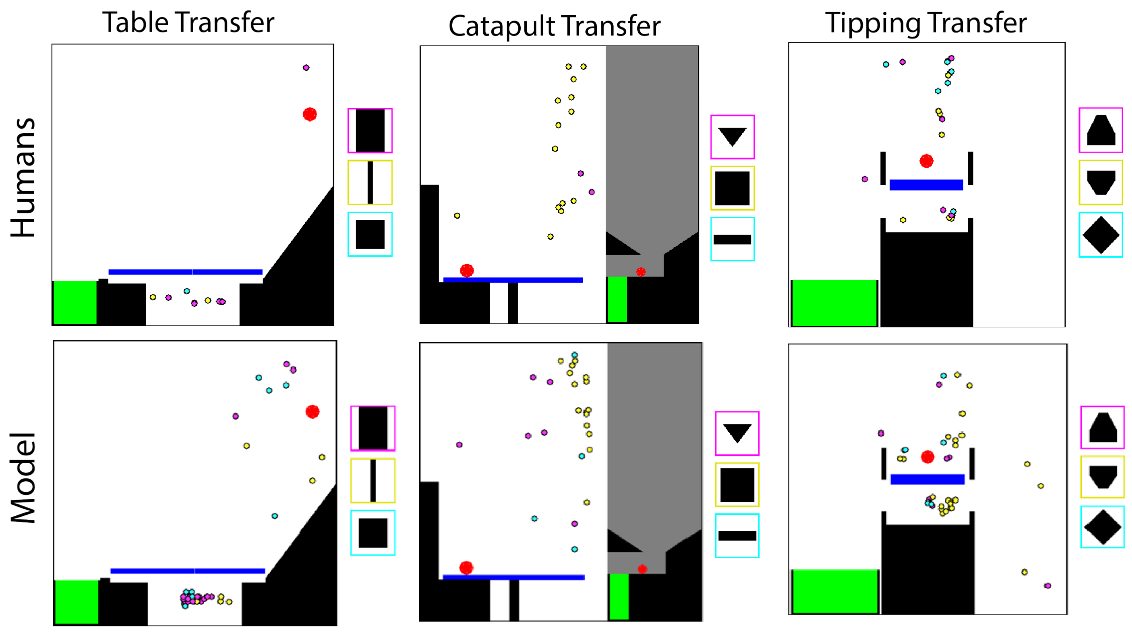


Figure 5-6: Performance in Experiment 2. (A) Proportion of successful participants on testing levels is generally higher for those trained on the strategy. (B) Specific placements and tool used on the first attempt for the transfer conditions in each strategy. People were trained on levels from Experiment 1.

RPP+SSUP results

Both the overall patterns in participant accuracy across training and test, as well as the more fine-grained placements used by participants on their first attempts, are captured by the Relational Policy Program model. RPP+SSUP similarly improves across training in Catapult

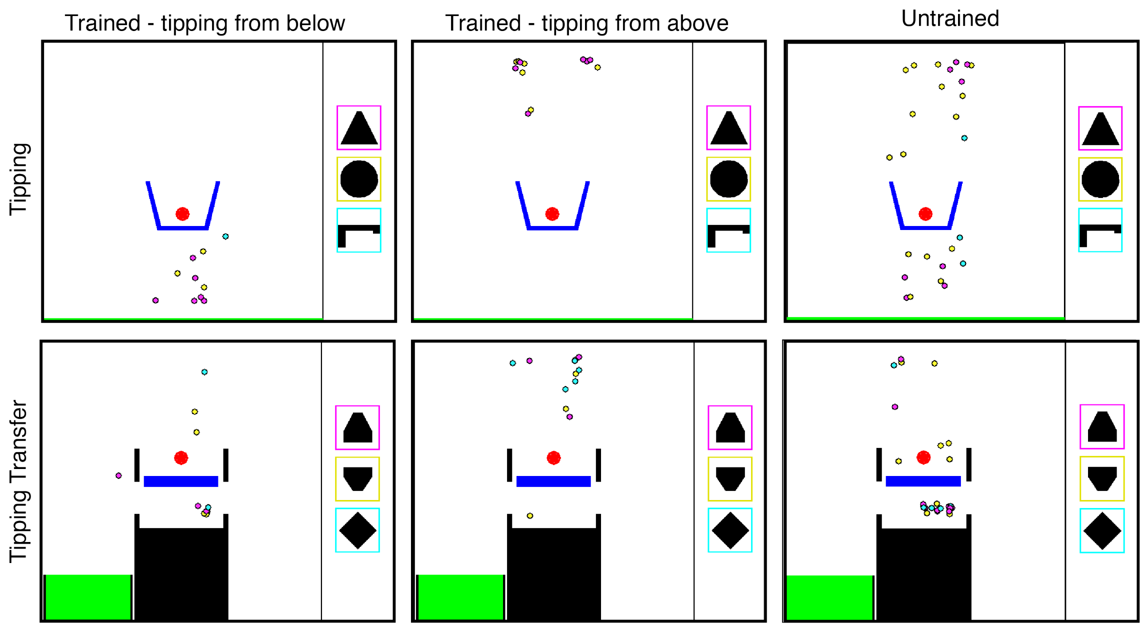


Figure 5-7: First placements of participants in the *Tipping* condition split into those who learned to tip from above, those who learned to tip from below, and those who were untrained.

and Tabling (see Figure 5-5), although it performs considerably better in Tipping from the first encountered level (see [5] for a discussion of this). RPP+SSUP similarly captures the pattern of results in testing performance – its accuracy on the Catapult and Table Transfer levels is significantly better when it is trained on the matched levels than when it is not. Two alternative models, one in which RPP is treated as a model-free mechanism and one in which only object-oriented strategies are learned rather than relational strategies, fail to capture this same pattern of results. In particular, both perform badly on the Catapult transfer level in testing, underscoring the importance of relational concepts rather than spatial object priors alone.

At a finer-grained level, we can also compare RPP+SSUP's first placements to human participants (Figure 5-6). RPP+SSUP is clearly capturing the essence of the correct strategy across all three transfer levels. In Table Transfer, RPP+SSUP is very likely to place a supporting object beneath the two planks, similarly to participants trained on Table. In Catapult Transfer, RPP+SSUP is clearly taking first placements that will act to catapult the ball. Similarly to humans, it is more likely to choose the largest object to do so, but not in all cases. Finally, in Tipping Transfer, RPP+SSUP shows a similar split to human participants in selecting a mixture of placements *above* and *below* the platform in order to try to get the ball into the goal.

Broadly, the successful transfer of strategies to different levels which maintain relational concepts but change the composition of the scene suggests that people flexibly adapt learned skills to new settings, and that this can be captured with relational programs over physical and spatial variables.

5.2.3 Experiment 3: Composing strategies and models

Based on Experiments 1 and 2, we established that people can learn strategies which shape how they interact with a new problem. Previous work has found that people can learn about object properties such as weight by observing how they interact with other objects [192, 244], but we do not know whether this kind of object property learning can be combined with learned strategies.

If people learn strategies that are more akin to *habits*, these new object types should not affect the kinds of actions that people take until they have acquired significant interaction experience with the new object type. In the more extreme case, if people learn about the object properties purely through observation with no direct interaction, their strategies could not be updated using most standard model-free RL learning procedures. Therefore, if people do compose observed models with learned strategies without any direct interaction experience, it suggests that people are using strategies to guide a model-based sampling process with an updated model, or that the representations of the strategies themselves are parameterized by a model. This presents a unique and complementary perspective on model-based and model-free decision making in humans, as prior work generally assumes models are learned by interacting with the world, not via passive observation.

Stimuli

We designed one new type of levels to test model-strategy composition, *Launch*, and included a subset of the *Catapult* levels from Experiments 1 and 2. In each level type, the ball size is kept constant so that participants cannot learn strategies that implicitly depend

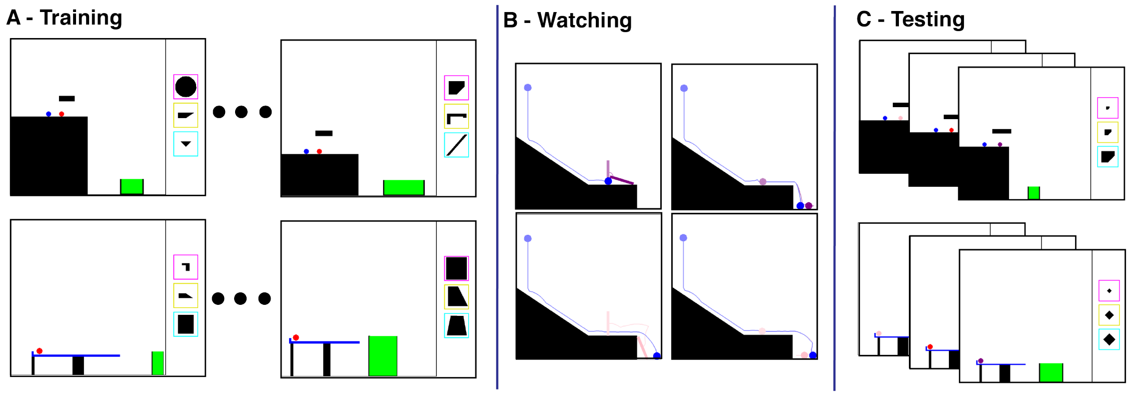


Figure 5-8: (A) Participants are trained on one level type. (B) Participants watch four short videos showing a blue ball interacting with pink and purple objects which have lower and higher density respectively. (C) Participants play a level in the same strategy category as training, but with a potentially light (pink) or heavy (purple) ball. They also play one randomly selected level from the other strategy types.

on the weight of the ball through its size.

In *Launch*, participants must drop a tool onto a ball that will roll into another ball and cause it to fall into the container. Depending on the relative positions and heights of the container and table, succeeding could require hitting the ball very hard or very lightly. Physically, we measure this as the amount of momentum that needs to be applied to the ball in order to succeed. In the training levels, the positions of the balls, height of the table, and position of the goal is varied.

Finally we also include a subset of the *Catapult* levels from Experiments 1 and 2. Depending on the location of the container relative to the ball, succeeding requires applying differing amounts of momentum to the platform to ensure the ball does not under or overshoot the container.

Procedure

Participants were trained on 5 levels of either the *Catapult* or *Launch* types, with 3 attempts per trial. After training, all participants watched two videos of a blue ball interacting with a pink object, and two videos of it interacting with a purple object. The pink objects were lighter (with a density 0.2× the blue and red objects), while the purple objects were heavier (with a density 2× the blue and red objects). After watching the four videos, participants then played a level of the same type as their training condition, but with either a red, pink or purple object. In this way, we could test whether people composed knowledge from observing dynamics with a learned strategy without requiring any interaction experience. Participants then also played every other level type with a randomly selected object mass.

At the end of the experiment, participants filled out a questionnaire which included the questions: “Which color corresponds to the heaviest objects?" and “Which color corresponds to the lightest objects?". Both questions could be answered with “red”, “purple” or “pink”.

We recruited a total of 168 participants across 9 conditions (3 category types $\times$ 3 weight conditions). Like in Experiments 1 and 2, we only analyzed data from participants who succeeded on at least one training level. Additionally, since we were interested in whether people composed new knowledge about object properties with learned strategies, we only included participants who either correctly answered which object color was heaviest (purple), or which object color was lightest (pink) to ensure they had learned relative object weights. Using both filters resulted in 31 participants being eliminated from the analysis (16 who failed because of the success criteria, and 15 because of the questionnaire).

Results

Participants showed improvement throughout training in all conditions (all $\chi^2 > 7.8$, all $p < 0.005$). In the *red* ball test trials, there is also an overall effect of training condition ($p = 0.01$), consistent with Experiments 1 and 2. To test model-strategy composition, we examine what people do on their first attempt in the test level when the density of the ball has changed.

In Figure 5-9A, we show participants’ first attempts in the heavy and light testing conditions for each level type. Participants appear to take the weight of the ball into account when choosing where to place an object and which object to place. For both *Catapult* and *Launch*, participants generally choose heavier objects and place them higher when the ball is heavier (purple).

We can quantify this by looking at the momentum applied across weight conditions for *Launch* and *Catapult* (Figure 5-9B).

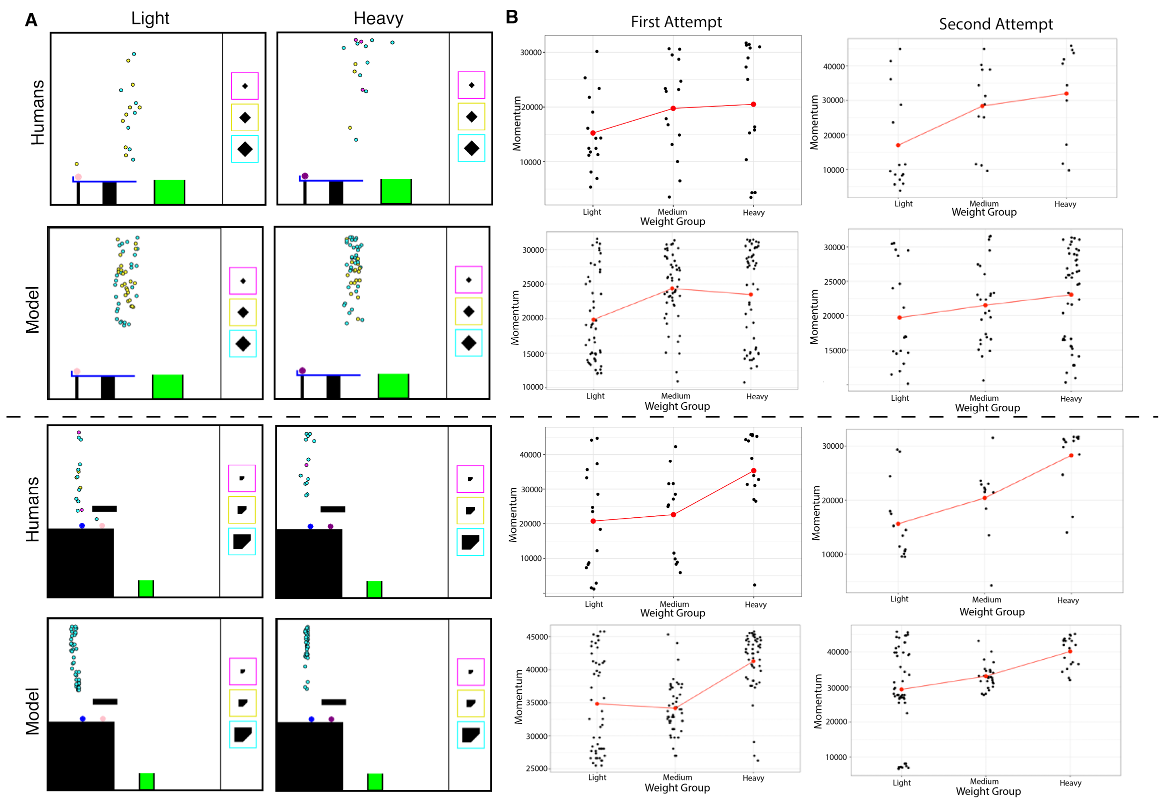


Figure 5-9: Performance in experiment 3. (A) First placements in the light and heavy conditions for catapult (top) and launch (bottom). (B) Momentum applied in each condition on the first attempt and second attempt for people and the model.

In *Launch*, we find that there is an effect of object weight on how much momentum participants impart with their tool ($F(2,40) = 5.0,\ p = 0.011$, partial $\eta^2 = 0.201$). We do not find a reliable difference for participants trained on *Catapult* ($F(2,43) = 1.6,\ p = 0.22$, partial $\eta^2 = 0.067$), but this appears to be because people in the heavy condition separate into two groups: one group which applies a significant amount of momentum, and one who does not. A subset of participants use the lighter tools on their first attempt, but after observing this single failure, 13/15 participants used the heaviest tool on their second attempt.

RPP+SSUP results

RPP+SSUP again captures both the fine-grained and overall patterns of human behavior in this model composition experiment. In both the Launch and Catapult levels, RPP+SSUP converges on a similar qualitative strategy to people based on its training, but also opts for more often using the *larger* tools and dropping them *closer* to objects when they are lighter, rather than using a lighter tool. This reflects the kind of strategy RPP+SSUP and people have learned: while the tool shapes might be radically different across levels, the variable that matters is the size, and larger sizes are more likely to lead to success in training. While qualitatively the first placements look similar between RPP+SSUP and human participants, the momentum calculated from these placements *also* looks remarkably well matched across both the first and second attempts.

In Launch, both RPP+SSUP and humans show a much greater difference in momentum between the medium and heaviest conditions on the first attempt, while in Catapult the more significant difference is between the medium and light conditions. This further underscores the appropriateness of RPP+SSUP for finding strategies to *guide search* rather than simply applying any action from the policy. By using the simulator in the loop, RPP+SSUP can be composed with newly inferred physical variables by simply using passive observations to make inferences instead of requiring active interaction.

5.3 Discussion

Through three exploratory experiments, we showed that in a physical problem solving task, people learn relational strategies that are object-oriented (Exp 1), can be transferred to contexts that differ visually but rely on similar relational principles (Exp 2), and can be combined in a zero or one-shot way with model-based variables such as weight (Exp 3). These results hint at a picture of rapid trial-and-error learning which focuses on relational *strategies* that can be used to guide a model-based sampling procedure towards promising regions of the problem space. A computational model which implements relational strategies as logical programs that guide a model-based sampling procedure is able to capture many of the overall patterns in human behavior, as well as much of the detail in the particular actions used.

There remain several open questions for future work to better understand the underlying representation of these strategies and how they are connected to model-based reasoning. Specifically, the kinds of strategies learned here are not the same as more traditional tool-specific affordances that have been a popular framework for thinking about tools in cognitive psychology [81], although they are certainly related. This was by design – in our experiments we explicitly randomized the tools available for each problem, and therefore the learned strategies had to be agnostic to any specific tool. However, we want to emphasize that we consider the objects here to still be tools, just unfamiliar ones. We see this as akin to a paper plate that can be re-purposed as a scoop, or a broom that can be re-purposed as a table leg. In future experiments, we would like to investigate whether people can learn object-specific strategies that dictate how a particular object can be used to accomplish a new objective.

In Exp 3, we saw a mixture of people who composed strategies and models without any experience, and those who required a single interaction with the scene to adjust their strategy. This suggests that even those who do not immediately generalize learn a strategy that is abstract enough to allow rapid updates. Further work could study the form of this representation and why individual differences exist.

Our experiments suggest that people's trial-and-error physical problem solving does not fit neatly into model-based reinforcement learning or model-free reinforcement learning. Instead, the content of people's strategies and how they know to apply them might be based on more abstract principles, allowing for broader generalization than previous studies of decision making would imply. However, these experiments are exploratory, and many open questions remain. Does the applicability of a strategy really depend on physical concepts, or is it the recognition of key objects and object-object relationships that appeared in a training level? How diverse can the training levels be? Can people learn multiple strategies simultaneously? Such questions present a wide and exciting space for future work to better understand the representations of physical problem solving strategies and how they are applied to new scenarios.

Production rule	Probability
Programs	
P $\rightarrow$ `is_object_type`(OBJ, OBJTYPE)	0.33
P $\rightarrow$ `compare`(Q1, Q2, OP, DIM)	0.33
P $\rightarrow$ `query_value`(Q, DIM)	0.33
Properties	
Q $\rightarrow$ `size`(OBJ)	0.25
Q $\rightarrow$ `area`(OBJ)	0.25
Q $\rightarrow$ `distance`(OBJ1, OBJ2)	0.25
Q $\rightarrow$ `location`OBJ)	0.25
Objects	
OBJ $\rightarrow$ `find_object`(OBJTYPE)	0.33
OBJ $\rightarrow$ tool	0.33
OBJ $\rightarrow$ object	0.33
Dimensions	
DIM $\rightarrow$ x	0.5
DIM $\rightarrow$ y	0.5
Object Types	
OBJTYPE $\rightarrow$ *dynamic*	0.33
OBJTYPE $\rightarrow$ *goal*	0.33
OBJTYPE $\rightarrow$ SHAPE	0.33
Shapes	
SHAPE $\rightarrow$ Compound	0.2
SHAPE $\rightarrow$ Ball	0.2
SHAPE $\rightarrow$ Box	0.2
SHAPE $\rightarrow$ Polygon	0.2
SHAPE $\rightarrow$ Container	0.2
Operators	
OP $\rightarrow <$	0.5
OP $\rightarrow =$	0.5

Table 5.1: The prior $p(f)$ over programs, specified as a probabilistic context-free grammar (PCFG).

Chapter 6

Meta-Strategy Learning by Embodiment

So far, this thesis has demonstrated that people can rapidly update policies by using structured priors with mental simulation to arrive at good solutions to physical problems quickly, and that the priors can be updated across problems by meta-learning relational programs which can further constrain search. This chapter examines whether even more abstract "meta-strategies" can be learned, such as how much an agent should rely on mental simulation as opposed to observing failed actions, from very diverse sets of experiences. If these meta-strategies are learned, what kinds of experience shapes them, and in what ways?

This chapter examines the potential roles of development and embodiment on meta-strategy learning for physical problem solving. We first show that generic experience of development has a significant impact on the kinds of strategies found by adults relative to children for physical reasoning. As they age, children become significantly better at finding solutions to disembodied tool use puzzles. By the time people reach adulthood, they also take significantly different kinds of actions to find solutions. To understand the effect of embodiment, we extend the study to children and adults born with limb differences (fewer than two intact arms and hands) in a matched sample to our typically limbed participants. Comparing the groups, there is a striking overall effect of handedness on meta-strategy: both children and adults born with fewer than two hands take a *significantly* longer time to think before acting, and as a result take *fewer* overall actions to reach solutions to physical reasoning problems in comparable amounts of time. This cannot be explained by differences in mouse control. Taken together, our findings suggest that differences in 'embodiment' drive the acquisition of different meta-strategies for balancing acting with thinking, while developmental experience affects typically and differently limbed children similarly, pushing them to try riskier actions as adults and become generally more adept tool users.

6.1 Introduction

Everyday experience is both constrained and enabled by the bodies we inhabit. Taller people can reach further, while stronger people can lift heavier objects without assistance. 'Embodied cognition' [232] suggests that such constraints play a fundamental role in shaping our cognitive and perceptual experiences broadly. Many versions of embodiment theory suggest that these effects reach beyond the types of experiences we have (e.g., imagining a scene is intimately tied to our visual experiences), and into the way that we reason about those experiences. In support of this view, researchers have shown that by manipulating people's bodies and the actions they have access to, they can similarly manipulate their perceptual [143, 2, 141] and cognitive capacities, such as decision making [94]. Here we take a different approach: we ask whether a *lifetime* of differences in embodied experience can affect the ways that people think about and plan to act in the world, even when their capabilities for action are made equal in the current moment.

Researchers have studied the effects of embodied history on cognition by investigating the perceptual and motor capabilities of individuals born with limb differences, as those individuals have lifelong differences in the way that they can interact with artifacts and objects that can be contrasted with people with typical limb development. However, the tasks used to study these capabilities often require judgments related to absent body parts, and therefore differences in behavior might be driven by differences in available information. For example, while people without two hands are slower to judge whether a picture is of a left or right hand [142], other studies highlight that individuals who were born without any hands show similar response biases as those with typically developed limbs, despite having profoundly different embodied experiences [222, 223]. People with missing hands, however, lack first-person visual experience of that hand, and therefore have a subset of the relevant experience that people with both hands do.

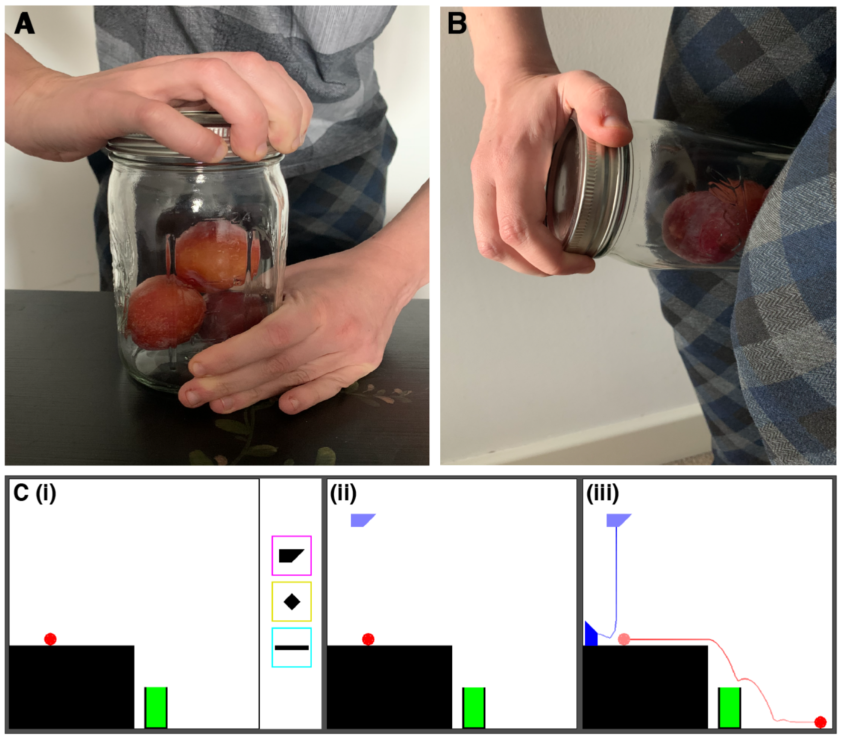


Figure 6-1: Having fewer than two hands changes the cost of interacting with everyday objects, like opening a jar. With two hands, opening a jar is easy (**A**) – one hand can stabilize the jar while the other one twists the lid. With a single hand (**B**), opening a jar is more difficult, but can be accomplished by using one's knees to stabilize the jar. The Virtual Tools game (**C**) equalizes action costs for individuals with different types of limbs by creating a virtual action space. (i) A participant selects a tool from the right-hand side of the screen and places it somewhere in the scene (ii). Once placed, physics is "turned on" and the participant can see the result of their action (iii); the blue line represents the observed object motion trajectories).

We are interested in whether long-term differences in embodiment might affect more general cognitive capabilities, even when the capabilities tested are divorced from particular body differences. We therefore study differences in "meta-strategies" for action, such as how persistent to be [131], how to adapt motor plans to one's own levels of motor variability and uncertainty [99, 67], or how to navigate the trade-offs between planning and acting [43]. Studies on these meta-strategies suggest that people make action decisions (both implicit and explicit) based on the expected costs and benefits of those actions [67]. Thus, if people with congenital limb differences have learned that actions are in general more costly – because, perhaps, it is more difficult to use artifacts designed for people with two hands (see Figure 6-1 for a common example of this involving jars) – we might expect that they will differ from two-handed people in how they select the actions they use, even when everyone is placed in a situation where action costs are equated.

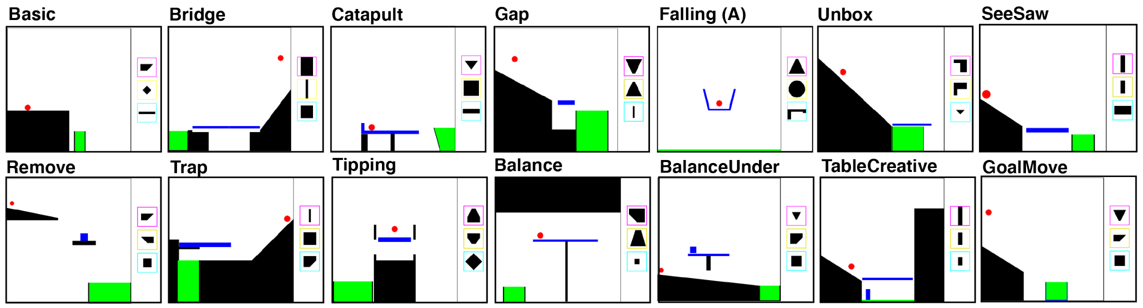


Figure 6-2: The fourteen levels of the Virtual Tools game [5] that participants played. These cover a wide variety of physical action concepts including "balancing", "launching", "catapulting", "supporting" and "tipping". To play the game, please see `https://sites.google.com/view/virtualtoolsgame`.

To test the influence of embodied experience on meta-strategy learning, we studied behavior in a *virtual* physical problem-solving task where all participants had equal capabilities to interact with the world, despite having different real-world embodied experience. We chose participant groups to represent a diverse range of embodied experience: typically developed children (5-10 yos) and adults, and age matched children and adults born with atypical limbs. Children have strictly less embodied experience than adults, while individuals with limb differences have dramatically different kinds of experience relative to individuals with typical limbs. By using a virtual task with simple manipulation inputs, we control for manipulation capabilities and instead study how embodiment affects the cognitive processes that support planning and reasoning more generally.

For the virtual physical problem-solving task, we use the recently introduced Virtual Tools game [5], which breaks the link between manipulation and physical problem-solving. The task requires people to use virtual objects as tools to solve a physical problem (such as knocking the red ball into the green goal, Fig. 6-1C) in a computerized environment using a single hand to control a mouse. Puzzles in the Virtual Tools game have the additional advantage of being equally unfamiliar to children and adults: all the 'tools' and puzzles are virtual and it is very unlikely for children or adults to have interacted with objects like these previously. Thus, unlike real-world objects which may be more familiar to adults than children, the use of the virtual tools should not be driven by direct experience with similar objects.

We initially expected that participants with congenital limb differences might solve tool use puzzles in more creative ways than two-handed participants, as they often must be more creative with their bodies to use real tools that are designed with bimanual use in mind (e.g., opening a jar by stabilizing it with their legs; Figure 6-1B). However, we did not find that children and adults with limb differences achieved different overall performance (solution rate or time until solution), but instead that they took fewer actions, using more time per action. This suggests that individuals with limb differences learn different meta-strategies for solving physical puzzles, and that these meta-strategies may be learned early in development.

6.2 Results

6.2.1 Equivalences in group characteristics

To ensure that the game equalized action capabilities between typically limbed (TL) participants and those with congenital differences in limbs (DL), and between children and adults, everyone completed a motor pre-test before proceeding to the main experiment (Fig 6-7). The motor pre-test consisted of ten trials where participants clicked on a star in the center of the screen, triggering the appearance of a circle in the periphery, which they were instructed to click on. We measured participants' reaction time as the time between the star and circle click, as well as position error, measured as pixels between their click and the center of the circle.

Children could control the cursor, with an average pixel error of $7.65px$ (95% CI=[6.74, 8.55]) and reaction time of $3.04s$ (95% CI=[2.67, 3.41]), albeit less accurately and more slowly than adults (error: $2.92px$, 95% CI=[2.52, 3.31]; RT: $1.91s$, 95% CI=[1.73, 2.09]). However, children's' control improved with development, reaching adult-like levels by the time they were 9-10 years old (see Fig. 6-3).

Importantly, typically- and differently-limbed individuals performed comparably in both motor error and in reaction time. While there was a difference in median click-

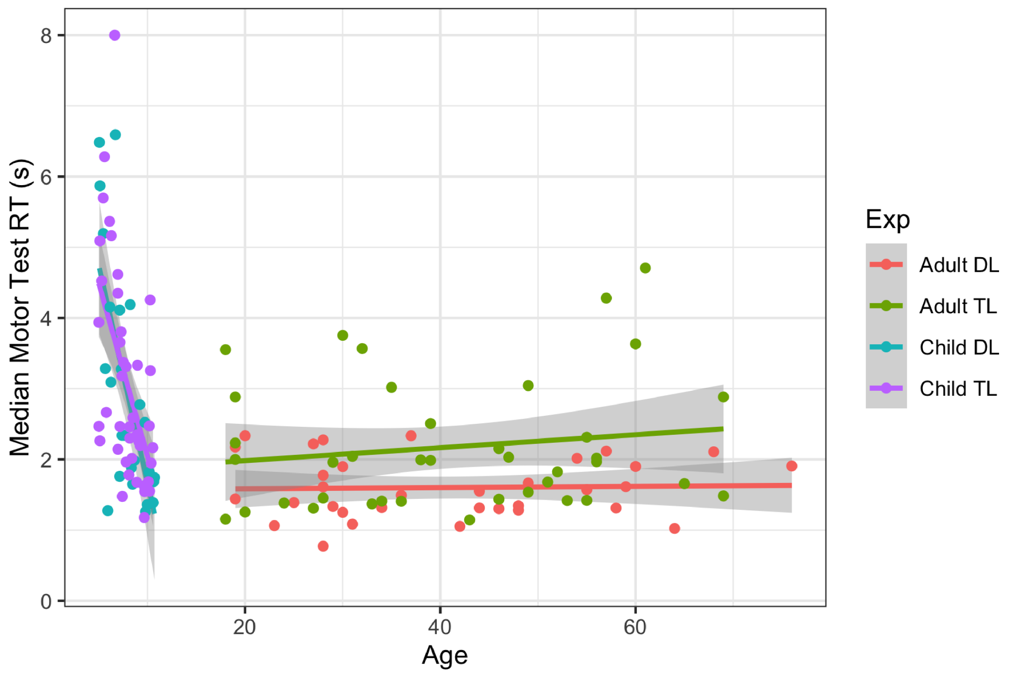


Figure 6-3: Performance on the motor task, specifically the motor reaction time compared with participant age across groups. Differently limbed (DL) participants were slightly faster on this task, suggesting that differences in mouse control cannot explain the additional time they take per action.

time between typically- and differently-limbed participants, differently limbed participants were slightly *faster* than those with typical limbs (by $356ms$, 95% CI=[14, 698]; $\chi^2(1) = 4.12,\ p = 0.044$), though typically limbed participants on average clicked marginally closer to the target (by $0.79px$, 95% CI=[-0.14, 1.72]; $\chi^2(1) = 2.78,\ p = 0.098$). There was no interaction found between age category and embodiment for either motor speed ($\chi^2(1) = 1.76,\ p = 0.19$) or error ($\chi^2(1) = 0.01,\ p = 0.98$). Note that the differences found in motor control are relatively inconsequential for the Virtual Tools game – $0.79px$ additional error would have little effect on $600x600px$ game screens, and an extra $356ms$ would be hard to detect with an average time between actions of over $10s$. Thus we find that our differently limbed participants had similar motor control capabilities to typically limbed participants, and so expect that both groups should have a level playing field for interacting with the Virtual Tools game; however, to control for any individual differences we include participants' median motor reaction time as a covariate in all performance analyses.

6.2.2 Contributions to overall performance

Given that our participant groups are matched on motor and cognitive measures, we next study whether overall performance on the Virtual Tools game differs across these groups. We measure overall performance in two ways: as the solution rate across all game levels (hereafter, accuracy), as well as the total time taken to solve those levels.

We find gross differences in accuracy between children and adults (adults: 85%, children: 77%; $\chi^2(1) = 11.2,\ p = 0.0008$), but no effect of hand embodiment ($\chi^2(1) = 1.21,\ p = 0.27$), nor any interaction between age and limb group ($\chi^2(1) = \ 0,\ p = 0.99$). We further find that treating age as a continuous variable additionally predicted success in a way that differed between these two groups ($\chi^2(2) = 12,\ p = 0.0025$), with children's accuracy improving with age (log-odds increase per year: 0.290, 95% CI = [0.038, 0.543]), and adults' performance getting worse (log-odds decrease per year: 0.029, 95% CI = [0.006, 0.053]).

The time to solve these levels also shows a similar pattern of results: overall, children are slower than adults to find a solution ($\chi^2(1) = 40.0,\ p = 2.6 * 10^{-10}$; Fig. 6-4), but we do not find evidence that typically- or differently-limbed participants are slower to solve the levels ($\chi^2(1) = 0.41,\ p = 0.52$), nor is there an interaction between limb group and age category ($\chi^2(1) = 0.22,\ p = 0.64$). However, we do find a similar pattern of how continuous age measures impact solution time: there is an effect that differed between children and adults ($\chi^2(2) = 44.5,\ p = 2.2 * 10^{-10}$), with adults slowing down with age (on average taking an additional $1.16s$ per year, 95% CI=[0.87, 1.46], $\chi^2(1) = 59.4,\ p = 1.3 * 10^{-14}$, and children becoming numerically but not statistically faster with age (on average taking $2.64s$ less per year, 95% CI=[-1.63, 6.90], $\chi^2(1) = 1.47,\ p = 0.23$).

Thus we find that development and aging cause noticeable changes in overall performance on the Virtual Tools game, but do not find that either typically- or differently-limbed participants are any better at the game. Nonetheless, participants might achieve an overall similar level of performance but do so in different ways; we therefore next consider whether typically- or differently-limbed participants might demonstrate differences in more detailed performance metrics.

6.2.3 Differences in embodiment

We first investigate whether there is a difference in the number of actions that typically- and differently-limbed participants took to solve each level, and hence the time between each action. The differently limbed (DL) participants took slightly fewer actions on average to come to a solution than the typically limbed (TL) participants (0.175 fewer actions, 95% CI=[0.024, 0.326]; $\chi^2(1) = 5.19,\ p = 0.023$; Fig. 6-4A). They also took notably more time for each action, including thinking more before the first action (3.79s more, 95% CI=[1.99, 5.59]; $\chi^2(1) = 17.0,\ p = 3.7 * 10^{-5}$; Fig. 6-4C), and between all subsequent actions (2.80s more on average, 95% CI=[1.51, 4.10]; $\chi^2(1) = 17.9,\ p = 2.3 * 10^{-5}$; Fig. 6-4D). Again, we found differences by age (number of actions: $\chi^2(1) = 6.51,\ p = 0.011$; time to first action: $\chi^2(1) = 13.4,\ p = 0.00025$; time between actions: $\chi^2(1) = 41.7,\ p = 1.1 * 10^{-10}$), but no evidence for an interaction between age and limb differences for any of these measures (number of actions: $\chi^2(1) = 2.07,\ p = 0.15$; time to first action: $\chi^2(1) = 0.10,\ p = 0.76$; time between actions: $\chi^2(1) = 0.16,\ p = 0.69$).[1]

Together, these results suggest that differently-limbed individuals learn a different meta-strategy for physical problem-solving. They learn to rely more on thinking about the problem and less on gathering information from their actions to solve physical problems.

[1]When testing for the effects of limb differences in children and adults separately, we find reliable differences in placement timing in both children (first placement: 4.13s, 95% CI=[0.96, 7.30], $\chi^2(1) = 6.52,\ p = 0.011$; between placements: 2.57s, 95% CI=[0.31, 4.84], $\chi^2(1) = 4.96,\ p = 0.026$) and adults (first placement: 3.29s, 95% CI=[1.32, 5.26], $\chi^2(1) = 10.7,\ p = 0.0011$; between placements: 2.63s, 95% CI=[1.26, 4.01], $\chi^2(1) = 14.2,\ p = 0.00017$). However, while we find that two-handed adults use more actions than adults with limb differences (0.286 additional placements, 95% CI=[0.047, 0.526], $\chi^2(1) = 5.49,\ p = 0.019$), two-handed children take numerically more actions, but this does not reach statistical significance (0.053 additional placements, 95% CI=[−0.140, 0.246], $\chi^2(1) = 0.29,\ p = 0.59$). Thus while we can claim that two-handed participants overall took more actions, we do not have enough evidence to discriminate whether this is because these differences are consistent, or whether the distinction grows through development.

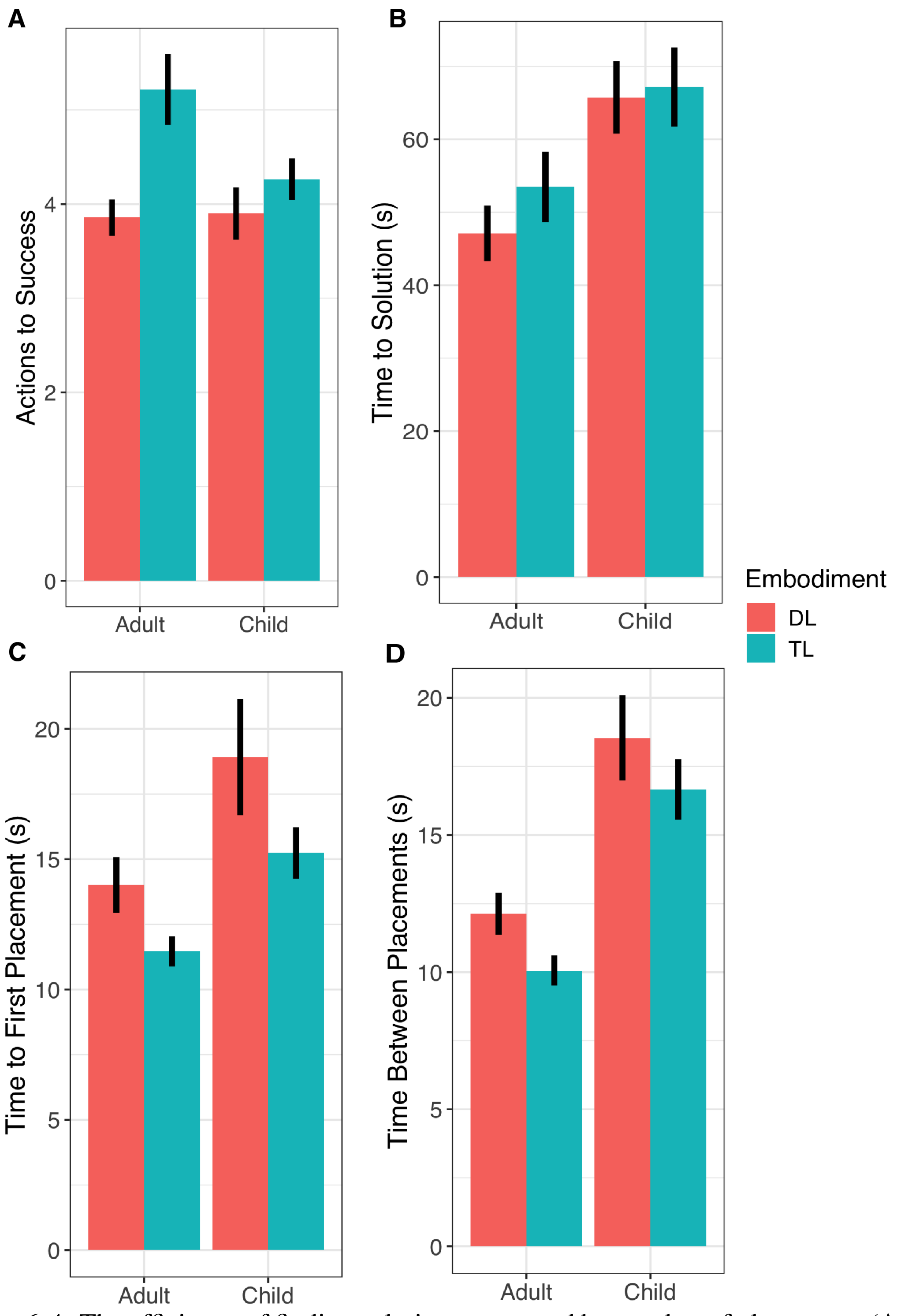


Figure 6-4: The efficiency of finding solutions measured by number of placements (**A**), and time as measured in seconds (**B**), time to first placement (**C**), and time between placements (**D**). Error bars represent 95% confidence intervals. Typically and differently limbed participants did not reliably differ on time to success, but differently limbed participants solved the levels in fewer placements, and took more time until the first placement and between placements.

We further investigated whether there might be differences in the types of actions selected between typically and differently limbed participants. As can be seen in Fig. 6-6, in many cases both typically- and differently-limbed participants often choose similar actions to solve the Virtual Tools game puzzles, though there are subtle differences. For instance, in the BalanceUnder level, participants with typical limbs (especially adults) were more likely to use a tool to hit the ball on the left instead of balancing the object on the center structure; perhaps differently-limbed participants are more likely to notice the balancing strategy as they are often required to balance objects using their residual limb when manipulating objects bimanually. We attempted to quantify these action differences using a similar methodology to [5]. However, just as we found in [5], this is a noisy measure that is not easily able to differentiate between groups, and therefore we found only subtle quantitative differences between typically and differently limbed participants; see Sec 6.4.4 for further details. Because this classification methodology is noisy, future work would be needed to discover whether there are noticeable differences in the types of strategies that typically- and differently-limbed participants consider.

6.2.4 Differences over development

Because we found a strong effect of age on overall performance, we consider how this is driven by changes in the actions that children and adults consider. In addition, although we did not find any evidence that the differences between differently and typically limbed participants on overall performance differed between children and adults, we test whether we can find more subtle differences in the actions that they choose.

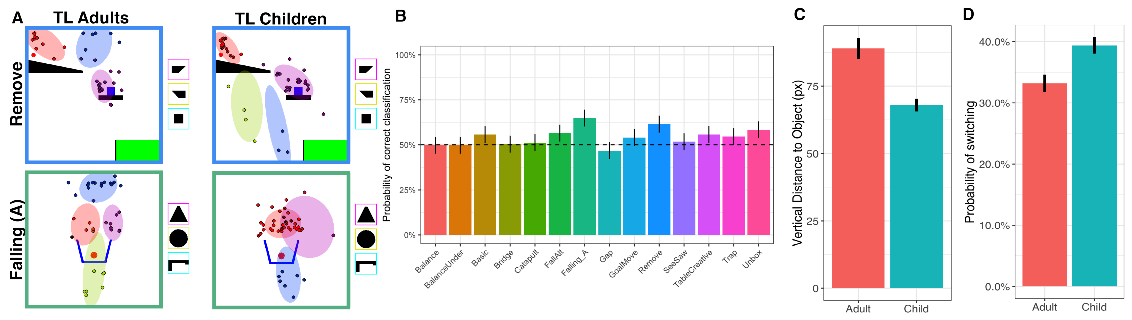


Figure 6-5: First placement comparisons across typically limbed (TL) children and adults. (**A**) An example of fitting a Dirichlet process mixture model to the first placements of adults (left) and children (right) for two tasks. (**B**) We tested whether we could classify participants' age group based on their first placement. Bars represent average classification accuracy across participants for each trial, with 95% confidence intervals on that estimate. (**C**) The average vertical distance between the tool placement and the nearest moveable object. Points represent the average for an individual participant, while bars represent the average across age categories. (**D**) The proportion of actions that belong to different placement clusters, indicating a "switch" in strategy.

First, we ask whether children's performance can be differentiated from adults even from the first action. We use the same leave-one-out classification used in [5], and find that over all levels, children and adults' first actions can be differentiated ($t(84) = 4.91,\ p = 4.5 * 10^{-6}$), and this classification is reliably above chance in 6 of 14 trials (see Figure 6-5B and Section 6.4.4 for details). Thus it is not just that children are taking more time and actions but choosing to do similar things – instead the way that children approach these puzzles differs from adults.

But what makes children's first actions different from adult's? In looking at those levels where children and adults' first actions were most distinguishable, it appeared that children were taking actions closer to dynamic objects in the scene. We tested this hypothesis by measuring the distance between first placements and dynamic objects in the scene, collapsing across trials. Analyzing these results, we found that when dropping tools from above or placing them below other objects, children tend to place the tools closer to the objects that they intend to move or support than adults do (average vertical distance in children: 74px, adults: 91px; $\chi^2(1) = 5.20,\ p = 0.023$; Figure 6-5C). We do not find evidence that this propensity changes with age in years ($\chi^2(2) = 1.59,\ p = 0.45$). This is not simply due to a tendency to place tools nearer to objects in general, as there is no reliable difference in horizontal distance between tools and the nearest object (children: $31.3px$, adults: $30.7px$; $\chi^2(1) = 0.65,\ p = 0.42$). It is therefore possible that these vertical differences might be driven by differences in real-world experience between children and adults, perhaps because of additional experience with falling objects, or because adults have better motor control when dropping real-world objects.

We also investigated whether there were any developmental differences in the ways that differently limbed children solved the levels relative to typically limbed children. We therefore used the same clustering and classification procedure, but instead of asking whether a participant came from their own or a different group, we measured whether each of the differently limbed children were better classified as a typically limbed child, or a differently limbed adult. If their actions resemble differently limbed adults, it would suggest that embodiment has an early effect in shaping the kinds of actions that individuals consider, but if their actions more closely resemble typically limbed children, then generic real-world experience might be the more critical factor in shaping the kinds of actions that individuals consider. As can be seen in Figure 6-6C, we find that differently limbed children are overwhelmingly better classified as typically limbed children than as differently limbed adults. Qualitatively, it appears that differently limbed children place tools close to other objects in the same way that typically limbed children do, unlike adults (Figure 6-6A).

We also investigated whether there was any difference in how children's actions evolved through the course of a single trial as compared to adults. We tested for differences in exploration behavior: would children be more or less likely to stick with similar actions to what they had just tried, or attempt something new? To measure this behavior, we used nonparametric clustering to group all actions across all participants within a single trial, in order to group actions into different "strategies" in a data-driven way (see Methods and Materials for details). We could then assign all actions to one of these strategies, and ask whether children or adults were more likely to switch strategies between actions. Children were in fact more likely to try new strategies than adults (children: 39% strategy switches, adults: 33%; $\chi^2(1) = 10.2,\ p = 0.0014$), suggesting that their lower accuracy might possibly be due to either exploring less efficient actions, or giving up on promising strategies early.

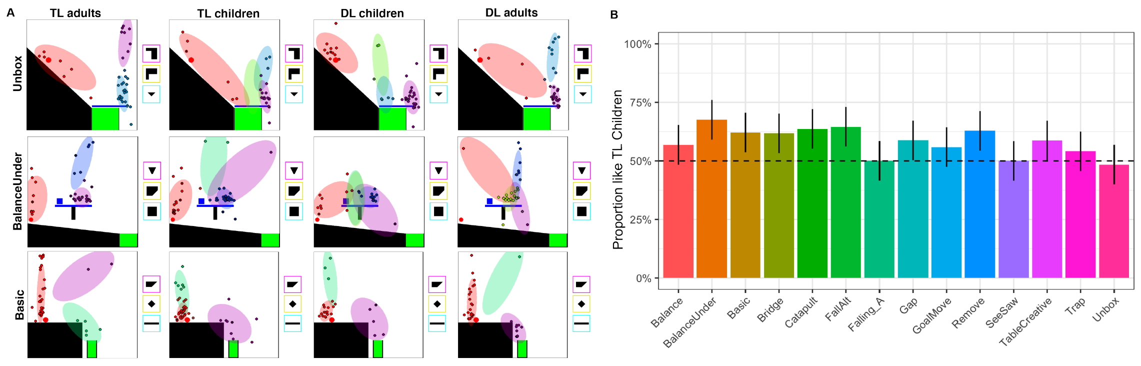


Figure 6-6: A comparison of differently limbed (DL) children, DL adults, typically limbed (TL) children and TL adults. **A** Examples of clusters determined for each group's first actions across three different level. **B** Classifying differently limbed (DL) children as either typically limbed (TL) children (closer to 100%) or differently limbed (DL) adults (closer to 0%). 95% Confidence Intervals are shown for parameter estimates of the mean.

6.3 Discussion

We asked participants – both with and without limb differences – to play the Virtual Tools Game to measure their skills in using tools in novel situations, and found that performance in this game increases throughout childhood, even though all children were above the age where they would be expected to be able to use tools in simple situations [118]. However, when we tested for the effects of embodiment on these capabilities, we did not find any differences in the development of these capabilities between congenitally differently limbed participants and typically limbed participants. We did, however, find differences in the way differently limbed participants solved these puzzles, relying more on *thinking* and less on *acting*, and that these differences were relatively consistent in both children and adults.

This has several implications for embodied cognition and tool use. We found minimal differences in the kinds of actions used by differently limbed individuals relative to typically limbed individuals across both children and adults, suggesting that the process underlying physical problem-solving is not dependent on similar kinds of manipulation experience. Instead, it appears that an understanding of how to manipulate the world develops similarly regardless of embodied experience, but that embodiment can set in place different general cognitive strategies for deploying this understanding.

Within embodied cognition, to our knowledge we give one of the first demonstrations of natural embodiment affecting a high-level decision making task unrelated to body or hand representations. Perhaps surprisingly, the way in which embodiment affects individuals does not take the form of affecting their specific action choices; rather, embodiment seems to affect the strategies people use to approach problems – that is, relying more on their simulations of the world rather than actions. We propose that individuals born with different limbs may learn at a very young age just how costly action can be; many actions that come naturally to their two-handed adults and peers might be very difficult for a differently limbed individual to execute and fixing a failed action might add even more cost. Naturally, this would lead to an increased reliance on planning before acting. What is striking is that this learned *meta-strategy* extends to a task in which actions are not any more costly – typically limbed and differently limbed individuals were well matched in their control of the mouse to play the game.

We suggest that these meta-strategies are learned through experience similarly to how people can learn to take more efficient motor actions over time [111]; however, the difference is that the target of learning here is not the motor plan itself, but when to deploy those motor plans. This would suggest that people with two intact hands should be able to learn similar meta-strategies with the appropriate cost structures. If the structure of the game were changed to make actions more costly in general – e.g., by requiring multiple clicks to select or place a tool – we would expect that all participants would learn to think more and act less. While these kinds of motor cost manipulations have been previously shown to affect which choices people make in visual discrimination tasks [94] and the efficiency of motor reaching actions [209], it has not been shown that they affect the meta-strategies that people employ. By focusing on these types of utility manipulations, future work could further explore how these meta-strategies develop.

This finding also suggests a connection between two different approaches to understanding human tool use. The "manipulation-based" approach, more closely aligned with embodied cognition, suggests that tool use is supported by sensorimotor knowledge related to tool manipulation [25, 26, 227, 86, 221], while the "reasoning-based" approach suggests that tool use is supported by physical knowledge which allows more generic physical planning and action [5, 166, 168]. These theories, supported by neuroimaging and lesions studies, have led researchers to suggest that there are distinct cognitive systems supporting different kinds of tool knowledge [165, 84]. However, these results suggest a connection between the two systems: by its virtual nature and novel objects, the Virtual Tools game must rely on reasoning-based systems for tool use, yet we find that manipulation capabilities affect how this reasoning is used. Thus the development of the reasoning-based system is grounded in the embodied way that we interact with the world.

Developmentally, our results extend existing knowledge about children's problem-solving and tool use. While even preschoolers [88] or infants [9] can understand cause and effect, our task involves reasoning about the specific effects of the virtual tools on their environment. Here studies of young children's tool use are informative. By 2-3 years old, children can use and select known tools [118]. By 4 years, children reliably copy adults' use of novel tools [27]. Yet until 8-9 years, children rarely innovate new tools, for example bending a pipe cleaner to use as a hook [17]. Recent explanations of this striking dissociation between tool use and tool innovation highlight the significant cognitive demands of innovation, including creativity, attentional control, inhibition, and planning [180].

Children's performance on our game is likely driven by some of these same cognitive skills. Like complex tool-use, or even tool innovation, the game requires children to explore a large solution space where they must strike an effective balance between exploration and exploitation [87]. On the one hand, children must avoid perseverating on a single solution [180]. The current study accords with work suggesting that perseveration is not high at this age [40], as well as theories that it may be predominantly a feature of the early motor system [205]. On the other hand, children must not explore the space too much, losing track of promising solutions as they go. Our results suggests that this is the more likely trap for children, who tended to switch solutions more often than adults, with lower success rates to match. We suggest therefore that the central difficulty with tool innovation and other physical problem-solving tasks at this age may precisely be the need to search through large solution spaces, where children's natural curiosity and propensity for exploration [171] may come at the cost of some short-term gains in solution-finding.

Our work opens up new directions for the role of embodiment and development in physical problem solving and tool use. By studying typically and differently limbed adults and children using an online game instead of motor manipulation experiments, we could more carefully investigate differences between populations at a very fine-grained level. Being born with a different body does not change the fundamental ways in which people try to act on the world, but it can change the kinds of meta-strategies people learn to plan and act efficiently. Even a lifetime of experience from childhood to adulthood does not affect those meta-strategies, even if the kinds of actions and the ways in which people modify their failed plans do.

6.4 Materials and Methods

6.4.1 Participants

We recruited a total of 145 participants across four groups: 40 typically limbed adults, 35 differently limbed adults, 45 typically limbed children, and 25 differently limbed children. Differently limbed (DL) and typically limbed (TL) participants were well matched for age (DL children mean: 7.91yo, sd: 1.84; TL children mean: 7.94yo, sd: 1.74; DL adult mean: 40.7yo, sd: 15.5; TL adult mean: 41.2yo, sd: 15.2). Adults were matched on education level, and we tested for similarities in cognitive capabilities by performing IQ tests on a subset of the children from both the typically and differently limbed groups. We assessed both Raven's matrices measures of spatial IQ [179], and BPVS as a measure of verbal IQ. 20/25 differently limbed children and 34/43 typically limbed children were tested. All scores were within normal range (lowest for BPVS 75, highest 134, lowest for Ravens 85, highest 135. Means 113 (differently limbed) 111 (typically limbed) for Ravens. Means (differently limbed) 105, (typically limbed) 108), and 2-sided unpaired t-tests show no evidence for differences between these populations (Ravens: $t(40) = 0.5,\ p = 0.6$, BPVS: $t(50) = 0.8,\ p = 0.4$).

Differently limbed children were recruited through the BOLDkids database of volunteer families and via Reach and Limbo (charities supporting children with upper limb differences and their families), while typically limbed children were recruited through a university-affiliated developmental cognition facebook page.

We excluded two differently limbed adults from analysis as they were amputees rather than congenitally differently limbed. We also excluded results from two typically limbed children: one due to a data recording error, and one who provided unreliable motor test data due to continuously clicking rather than attempting the task.

6.4.2 Experiment

The experiment progressed through two stages main: motor pre-test, and Virtual Tools game. After each of these phases was a short questionnaire.

All participants were given the same experiment, with only three exceptions that differed between children and adults: (1) children received simplified instructions for all phases, (2) adults played one additional Virtual Tools level that we removed from the children's experiment due to excessive challenge, and (3) adults were given a more extensive questionnaire that included additional questions about the strategies they had used and video games they had played before.

Motor pre-test

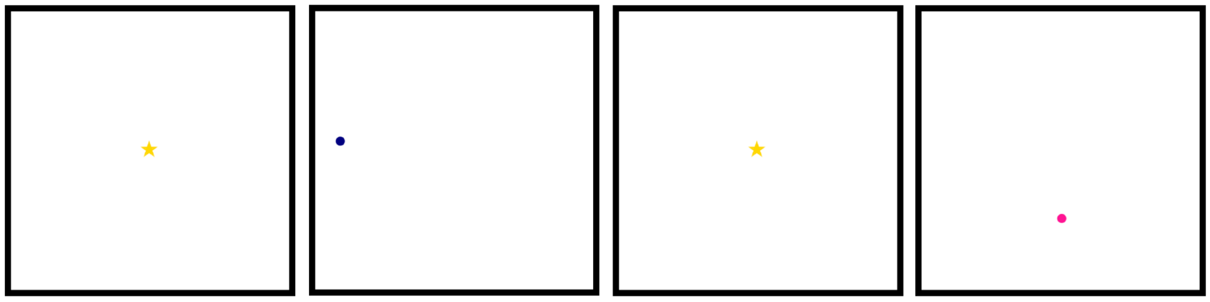


Figure 6-7: An example of two rounds of the motor pre-test. Participants clicked first the star, then a circle in the periphery, as quickly as possible.

The motor pre-test (Figure 6-7) was used to measure participants' facility with controlling the mouse cursor. In this phase, each trial would begin with a star in the center of a 600x600px area on the screen. Once the star was clicked, a circle of radius 10px would appear randomly either 150px or 250 px from the center of the screen, and participants were instructed to click on the circle as quickly and accurately as possible. However, when the circle was visible a click anywhere on the screen would end the trial, so that we could appropriately capture speed-accuracy trade-offs without worrying that people might not notice mis-clicks. Participants all completed 10 motor test trials, with five circles appearing 150px away, and five appearing 250px away.

On each trial we measured (a) the time between the circle appearing and when the participant clicked on it, and (b) the distance (in px) between the center of the circle and the mouse click. As a measure of participants' motor facilities, we took the median of both of those measures across all 10 trials; we used the median to avoid skew from outlier trials (e.g., if the participant accidentally clicked or was distracted on a trial), and found in pilot testing that this was a relatively stable measurement.

Virtual Tools game

Following [5], at the beginning of this phase, participants were given instructions about how the game functioned, then three introductory trials: one which required them to place tools without an objective, and two simple levels that they were required to solve but that were not analyzed.

Following this, participants would be given the analyzed trials to solve. In each of these trials, there would be a goal condition (e.g., "get the red object into the green goal area") that they needed to accomplish by placing a single tool somewhere on the screen such that it did not overlap with other objects or illegal areas. Participants could attempt to place tools as many times as they wished, but were required to reset the world back to its initial state between attempts. Participants could move onto the next trial once they had accomplished the goal, or 60 seconds had passed. On each trial, we recorded the actions that each participant took (which tool they selected, and where they placed it) and at what time, as well as if they had solved the level or not.

Adult participants were given 15 different trials to solve: the 14 shown in Fig 6-2, and one additional level: Spiky (see [5]). Because of the low solution rate of adults on this level (18%), we were concerned that it might frustrate children and cause attrition, so only had children play the other 14 trials, and removed the Spiky level from analysis.

Questionnaire

After both the motor test and Virtual Tools task, participants were given a short questionnaire. After the motor task we asked what device participants were using to control the mouse, and, for differently limbed participants, whether they were using their intact or non-intact limb. After the Virtual Tools task, we asked whether participants had changed how they were controlling the mouse. Additionally, for the adult participants we included the questions asked in [5], which asked about participant age, gender, prior video game experience, and free-form responses about strategies they had used on the task.

Statistical methods

For aggregate performance analyses, we analyzed data at the trial level, using summary statistics including (a) whether the trial was solved, (b) when the last action was taken to produce a solution, and (c) how many actions were taken over the course of the trial. For the latter two metrics, we conditioned our analyses only on *successful* trials, as we were interested in the mental processes that led to solutions, and not processes that might be indicative of frustration or perseverance.

We modeled all statistical analyses as (generalized) linear mixed effect models using the 'lme4' package in R [14]. We treated accuracy as a binomial response, time to solution as having Gaussian error, and placements as a Poisson process (using the number of non-solution placements as the dependent variable so that we could observe zero-placement outcomes). In all models, we assumed **random intercepts** for **participants** and **trials**.

Additionally, we included two covariates in our analyses. For all analyses, we used the median motor test response time as a covariate, as we had hypothesized that motor facility might cause better performance. We selected response time instead of error because the two measures were somewhat correlated ($r = 0.34$), and in pilot analyses we found that adding a second motor measure explained very small amounts of additional variance in performance over just a single measure. Finally, because we observed such large effects of age on performance, for all analyses testing the difference between differently- and typically limbed participants, we included age as a covariate, allowing its effects on performance to differ for children and adults (treated as an age by child/adult interaction).

6.4.3 Participant Demographics

Please see Tables 6.1 and 6.2 for participant demographic information.

Group	Handedness			Input Device		
	Left	Right	Ambi.	Mouse	Touchpad	Other
DL Adults	0.45	0.55	0	0.61	0.36	0.03
TL Adults	0.08	0.92	0	0.55	0.45	0
DL Children	0.36	0.64	0	0.32	0.68	0
TL Children	0.07	0.89	0.04	0.50	0.50	0

Table 6.1: Demographics summaries for participants. DL = Differently limbed, TL = typically limbed.

Age Category	Affected Limbs		
	Right Arm	Left Arm	Both
Children	0.32	0.44	0.24
Adults	0.42	0.55	0.03

Table 6.2: Proportion of differently limbed adults and children who have an affected right or left arm, or both arms.

6.4.4 Differences in action choices

Methodology

To test differences in action choices between groups, we apply a leave-one-out classification analysis similar to [5]: for each participant, we form probability distributions as

Dirichlet Process Mixture models over the actions taken by all other members of their embodiment category and all actions taken by members of the other category, then calculate the relative likelihood that the tool placement in the first attempt for a given level was a member of the correct group. If this measure is on average reliably above chance on a trial, this suggests that typically and differently limbed participants are beginning their solution search in different ways.

For analyses involving switching behavior, we aggregate all data across all groups and apply a Dirichlet Process mixture model on the spatial component of actions (where tools were placed) to obtain a single consistent clustering for all participants across all groups.

Differences between TL and DL participants

Overall, we found a trend towards being able to classify typically and differently limbed adults (mean classification accuracy: 51.4%, 95% CI=[49.9, 52.9]; $t(71) = 1.91$, $p = 0.060$), but found only two tasks where the confidence interval on the estimated classification probability exceeded chance: BalanceUnder and Bridge. Both of these tasks require preventing objects from falling – either by placing a tool as a counterweight to another object or preventing two platforms from falling inwards – which might be a strategy the differently-limbed participants used more often in daily life, as they are have to rely on their residual arm (which is shorter than their intact arm) when manipulating objects bimanually. However, given that the estimated classification probability is not far from chance and that we cannot reliably classify actions in other trials that rely on balancing, future work would need to investigate particular differences in strategies learned from interaction with the environment.

6.4.5 Additional tests for moderators

Here we test for the effect of other potential explanatory variables on our set of metrics describing performance on the tools game, including (1) participant gender, (2) the type of device participants used to control the game (mouse or touchpad), (3) whether participants were dominantly left- or right-handed, and (4) for the DL participants, what kind of limb differences they had. While we find possible impacts of these variables on overall performance, in no cases do we find any interactions between these variables and embodiment, suggesting that they should not impact the main results of the paper.

Gender

We exclude 8 participants from analysis (3 DL adults, 5 TL adults) due to recording errors.

There is a main effect of gender on accuracy ($\chi^2(1) = 9.88,\ p = 0.0017$), with males slightly outperforming females (83% vs 77%), but no interaction with embodiment class ($\chi^2(1) = 0.53,\ p = 0.47$). We also find a small effect of gender on the first action time (males: $13.8s$, females: $15.2s$, $\chi^2(1) = 5.56,\ p = 0.018$), but again no interaction ($\chi^2(1) = 0.75,\ p = 0.39$. Beyond this we find no evidence for any main effects on our main performance variables (actions: $\chi^2(0.33) = 1,\ p = 0.57$, solution time: $\chi^2(1) = 0.32,\ p = 0.57$, time between actions: $\chi^2(1) = 1.19,\ p = 0.28$), nor any interactions between gender and embodiment (actions: $\chi^2(1) = 0.08,\ p = 0.77$, solution time: $\chi^2(1) = 0.54,\ p = 0.46$, time between actions: $\chi^2(1) = 0.00,\ p = 0.95$). Although there may be slight differences in how males and females perform this task overall, because we found no interactions with embodiment, the effect of embodiment itself does not depend on participant gender.

Input device

We excluded one DL adult participant who did not use a touchpad or mouse for controlling their computer.

We find a main effect of device on time to solution ($\chi^2(1) = 7.96,\ p = 0.0048$), with participants using a mouse solving the levels slightly faster than participants using a touchpad ($53.2s$ vs $63.8s$). However, we found no interaction between device and embodiment on solution time ($\chi^2(1) = 0.47,\ p = 0.49$), nor did we find any other main effects of device (accuracy: $\chi^2(1) = 0.04,\ p = 0.84$, number of actions: $\chi^2(1) = 3.33,\ p = 0.068$, first action time: $\chi^2(1) = 2.43,\ p = 0.12$, time between actions $\chi^2(1) = 1.96,\ p = 0.16$) or interactions between device and embodiment category (accuracy: $\chi^2(1) = 0.00,\ p = 0.96$, number of actions: $\chi^2(1) = 1.12,\ p = 0.29$, first action time: $\chi^2(1) = 0.14,\ p = 0.7$, time between actions $\chi^2(1) = 0.32,\ p = 0.57$). Thus participants' choice of device should not impact our tests of the effects of embodiment.

Hand laterality

We exclude 1 TL child who was ambidextrous.

We find no effect of hand laterality on any of our dependent variables – neither main effects (accuracy: $\chi^2(1) = 0.01,\ p = 0.91$, number of placements: $\chi^2(1) = 0.35,\ p = 0.55$, solution time: $\chi^2(1) = 0.00,\ p = 0.98$, first action time: $\chi^2(1) = 3.12,\ p = 0.077$, time between actions $\chi^2(1) = 0.78,\ p = 0.38$), nor interactions between laterality and embodiment category (accuracy: $\chi^2(1) = 0.05,\ p = 0.83$, number of placements: $\chi^2(1) = 2.00,\ p = 0.16$, solution time: $\chi^2(1) = 0.84,\ p = 0.36$, first action time: $\chi^2(1) = 0.80,\ p = 0.37$, time between actions $\chi^2(1) = 0.05,\ p = 0.83$).

Chapter 7

Ongoing and Future Directions

People are perhaps uniquely adept at using intuitive theories not just to make inferences, but also to *act* and *intervene* on the world in order to accomplish their goals (Section 2). Two mostly separate bodies of work on human decision making have focused on either how people represent structured problems in ways that support efficient search (Section 2.2.1) or how people learn in unstructured settings from many trial-and-error experiences (Section 2.2.2). There have been fewer attempts to grapple with the much more complex question: how do people rapidly learn from trial-and-error experience within structured problems in a way that supports efficient search and flexible transfer?

This thesis shows how to inject structure in the forms of objects, relations and physics into trial-and-error learning to improve both the flexibility and efficiency of human and machine problem solving. The presented framework, Sample Simulate Update (SSUP), gives simulation a central role in considering how people act and learn so efficiently, but suggests that simulation alone is not enough. Structured action priors, centered around objects and relations, play an equally important role in shaping the landscape of *how* people use a simulator effectively to find solutions to problems. This presents a new path forwards for human decision making by combining components that have been influential in problem solving and reinforcement learning separately into a single computational framework.

This body of work focuses on physical problem solving, in part because of the widespread interest in tool use and tool cognition across developmental psychology, anthropology, comparative cognition, and motor planning and learning. As many have argued [118], tool use remains an unparalleled window for studying intelligence across both animals and machines (2), and in particular the cognitive challenge of problem solving. This thesis takes a major step to providing a grounded computational account of how this kind of learned physical problem solving unfolds, and presents a new domain for studying physical problem-solving at a fine-grained, quantitative level to investigate predictions of various models.

7.1 Summary of main results

As a first step towards more flexible machine agents in the domain of physical problem solving, Chapter 3 introduced the Gluing Task which required agents to stabilize a tower of blocks by applying as little glue as possible. This task proved very difficult for an unstructured deep learning agent. However, by incorporating *relational and object structure* into the action space, the agent was able to outperform human participants on the task and even generalize to larger towers than it experienced during training.

Moving towards more general physical problem solving, Chapter 4 introduced The Virtual Tools Game: a virtual 2D game inspired by human and animal tool cognition. By developing a virtual platform, tool use was studied not just from a *qualitative* perspective, but from a *quantitative* perspective as well. Across 30 unique levels, the game tested a variety of different physical concepts such as "catapulting", "tipping", "supporting" and "blocking" objects. While this thesis focused on tool *use*, the game is trivially extensible to problems such as tool *innovation*, and even tool *modification*. To understand how humans solve problems based on very limited trial-and-error experience, the "Sample, Simulate, Update" (SSUP) framework was proposed, consisting of object-oriented action priors ("Sample"), a noisy physical simulator ("Simulate"), and an update mechanism ("Update") to guide search towards more useful parts of action space based on the results of both simulations and actions. SSUP captured human behavior across multiple levels of granularity: from human accuracy across different levels, to the particular placements used on the first and last actions within levels. By contrast, methods that did not incorporate the object-oriented action prior, simulator, or update mechanism were unable to explain human behavior in the domain. While SSUP has clear limitations, the general framework of learning to search in a model-based way has significant potential for explaining the flexibility and efficiency of human trial-and-error learning.

People are not just capable physical problem solvers, they are highly efficient physical strategy learners. Chapter 5 focused on how action priors could be *learned* and *transferred* to enable even more efficient search for problems that people have previously experienced. Relational program policies (RPP) provided a means of learning structured, relational action priors across tasks by representing them as relational *programs* drawn from a probabilistic grammar. Replacing the "Sample" component of SSUP with RPP was sufficient to explain how people learn relational strategies from just a handful of related levels, generalize those strategies to tasks that look visually distinct from training levels, and even compose these learned strategies with new physical variables from only one or two attempts.

Trial-and-error experience does not just happen within the span of a psychology study – people try and fail to solve problems every day in real life. Chapter 6 investigated how this *naturally embodied* trial-and-error experience might affect the meta-strategies people apply to physical problem-solving in a new context. Differently and typically limbed children and adults were recruited to play a selection of tasks from the Virtual Tools game to assess whether differences in naturally embodied experience would affect disembodied physical problem-solving. Dissociable effects of embodiment and development were found, with development affecting the *kinds* of actions children took relative to adults, and embodiment affecting the *meta-strategies* individuals used to solve problems. In particular, differently limbed individuals spent more time *thinking*, and less time *acting* relative to typically limbed individuals, even from a very young age despite minimal differences in motor capabilities. This could be due to learned *costs* associated with action – differently limbed individuals may learn to rely more on their simulations when considering a course of action, because trying and failing is more costly than for typically limbed individuals.

7.2 Towards discovering structure

This thesis mostly examined settings where the structure of the problem was known in advance, and investigated ways of taking advantage of that structure to improve flexibility and generalization within the domain of action. However, there are many settings in which the structure of the problem may not be known in advance, and must simultaneously be discovered by either watching others, or by interacting with the world. Chapter 5 showed some first efforts in this direction, discovering structure through relational programs that best distinguish successful and unsuccessful actions, but this still assumed access to a relational, probabilistic grammar. How might such structured representations be learned from experience?

At least two components are likely required: one component for *representation learning* to take unstructured information and find an embedding which is useful for a variety of tasks (broadly the task of *perception*), and one component for converting such representations into some actionable, structured form. As a first step in this direction, we tackled *structured perception* in the form of finding taxonomies of perceptual forms to improve classification in settings where very little data is available (often called "few-shot learning") [4]. To further move towards more human-like structured perception, our method also handled semi-supervised learning and fully unsupervised inference. Our insight was to combine Bayesian nonparametric inference techniques with deep representation learning in a single end-to-end framework that could simultaneously discover how to *adaptively* represent both simple and complex data distributions for few-shot learning.

This adaptivity is particularly important in few-shot learning, where both underfitting and overfitting are common problems, because current models are fixed in their capacity. To give an example, consider the problems of character and alphabet recognition in the Omniglot dataset [129]. Recognizing characters is fairly straightforward: each character looks alike, and can be represented as a single prototype (a uni-modal Gaussian distribution). Recognizing alphabets is more complex: the uni-modal distribution assumption could be violated, and a multi-modal approach could better capture the complexity of the distribution. Figure 7-1 shows a prototypical network [203] embedding for alphabets with

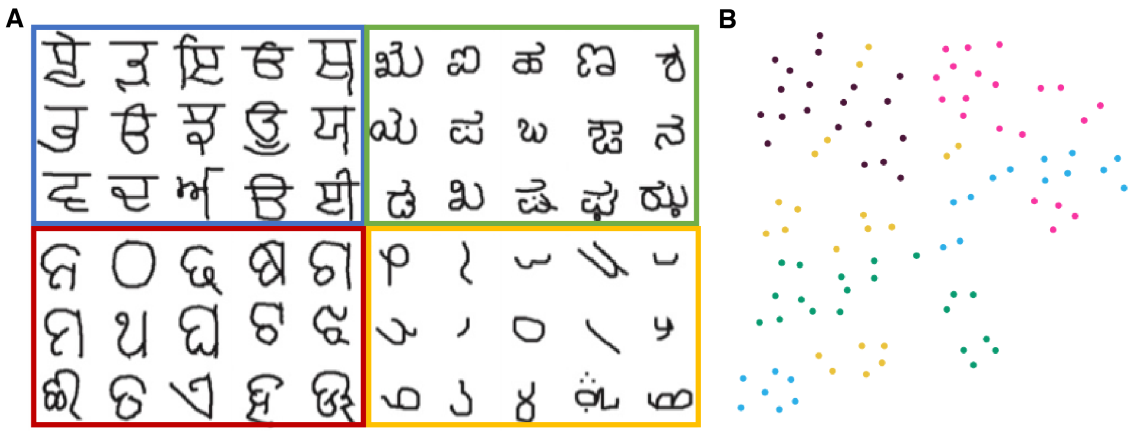


Figure 7-1: t-SNE visualization of the deep embedding from a prototypical network (right) trained for alphabet recognition on Omniglot (left). Each point is a character colored by its alphabet label. The data distribution of each class is clearly not uni-modal, in violation of the modeling assumption for existing prototypical methods, causing errors. Our infinite mixture prototypes represent each class by a *set* of clusters, and infer their number, to better fit such distributions.

this very issue. Even though the embedding was optimized for uni-modality, the uni-modal assumption is not guaranteed on held-out data.

Infinite mixture prototypes (IMP) combine deep representation learning with Bayesian nonparametrics to overcome this issue, representing each class by a set of clusters, unlike existing prototypical methods that represent each class by a single cluster. By inferring the number of clusters, infinite mixture prototypes interpolate between nearest neighbor (exemplar-based) and prototypical representations in a learned feature space, which improves accuracy and robustness in the few-shot regime. We showed the importance of adaptive capacity for capturing complex data distributions such as super-classes (like alphabets in character recognition), with 10-25% absolute accuracy improvements over methods that only represent clusters as the mean of the distribution, while still maintaining or improving accuracy on standard few-shot learning benchmarks. By clustering labeled and unlabeled data with the same rule, infinite mixture prototypes achieved state-of-the-art semi-supervised accuracy, and could perform purely unsupervised clustering, unlike existing fully- and semi-supervised prototypical methods.

Infinite mixture prototypes were inspired by the study of categorization in cognitive science. Exemplar theory [164] represents a category by storing its examples. Prototype

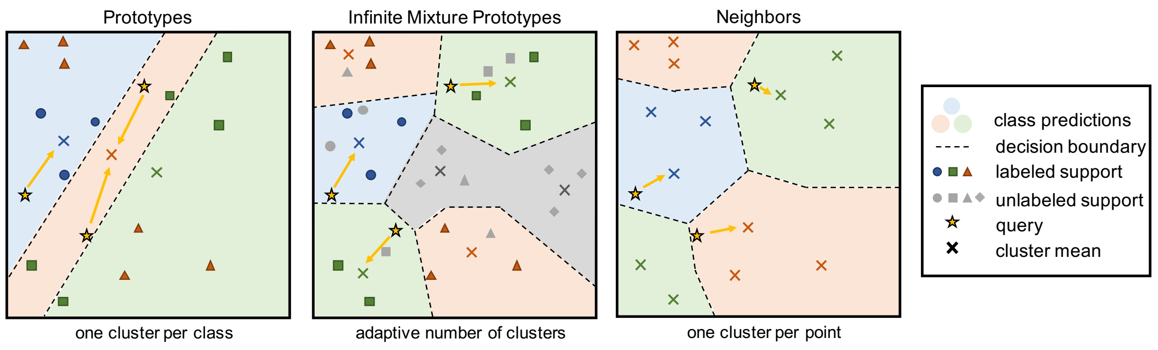


Figure 7-2: Our infinite mixture prototypes (IMP) method combines deep representation learning with nonparametric clustering to represent each class by a set of clusters in a learned feature embedding. The number of clusters is inferred from the data to adjust modeling capacity. IMP is optimized end-to-end to cluster labeled and unlabeled data into multi-modal prototypes.

theory [182] represents a category by summarizing its examples, by for instance taking their mean. [224] recognize that exemplars and prototypes are two extremes, and define intermediate models that represent a category by several clusters in their varying abstraction model. However, they do not define how to choose the clusters or their number, nor do they consider representation learning. [90] unify exemplar and prototype categorization through the hierarchical Dirichlet process to model the transition from prototypes to exemplars as more data is collected. They obtain good fits for human data, but do not consider representation learning. IMP aimed to take these insights towards machine learning, but could also be applied to the study of how humans simultaneously learn concepts as either exemplars or prototypes, spanning the space in between. We imagine this as a fruitful direction for future work in the space of structured perception for cognitive science.

Infinite Mixture Prototypes for action

To move towards learning structured representations for actions, we took some first steps towards applying infinite mixture prototypes to the problem of *skill* or *mode discovery*. Inspired by Gibson's notion of affordance, we imagine that for any given object, there are multiple ways an agent could interact with it in order to achieve different goals. For example, consider the domains shown in Figure 7-3. In both cases, a robotic agent represented as a gripper or hand must interact with the blocks in the scene in order to accomplish a

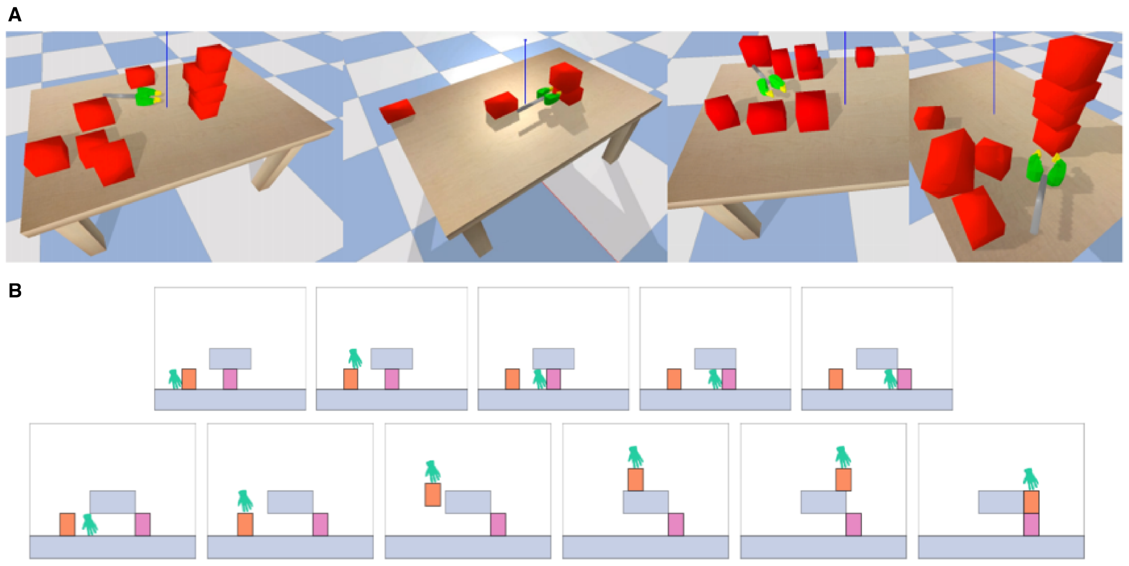


Figure 7-3: Domains for affordance learning in simple robotics settings. Top: a 3D environment, Figure courtesy of [238], where a robotic gripper must push a tower of blocks so that it moves into a particular region of space. Bottom: a 2D environment, Figure courtesy of [137], where a robotic gripper can push or pick up blocks in order to satisfy different kinds of specified relational goals.

specific goal.

In the top panel, the robot can push in any direction in order to move a stack of 3 blocks into a particular region of a table (Figure credit [238]). However, some pushes are *stable* in the sense that they won't topple the block tower, while others are *unstable* – those actions will cause the stack to fall. It would be helpful for an agent to realize that there are these two kinds of pushes that can be made, as only stable pushes will be helpful for actually achieving the specified goal.

In the bottom panel, the robot can take two kinds of actions on the blocks: picking them up (second row) and pushing them (top row). Again, these are two kinds of affordances that the block enables for the robotic agent, and both can be used together in order to solve complex problems. But how do we discover these different kinds of affordances?

The task of affordance learning in these problems can be related to the perceptual clustering problem presented previously: we must simultaneously learn *representations* of the actions for the robot and the blocks that allow us to predict how an action will affect the scene, and we must learn *how many kinds* of actions there are. This is essentially the same

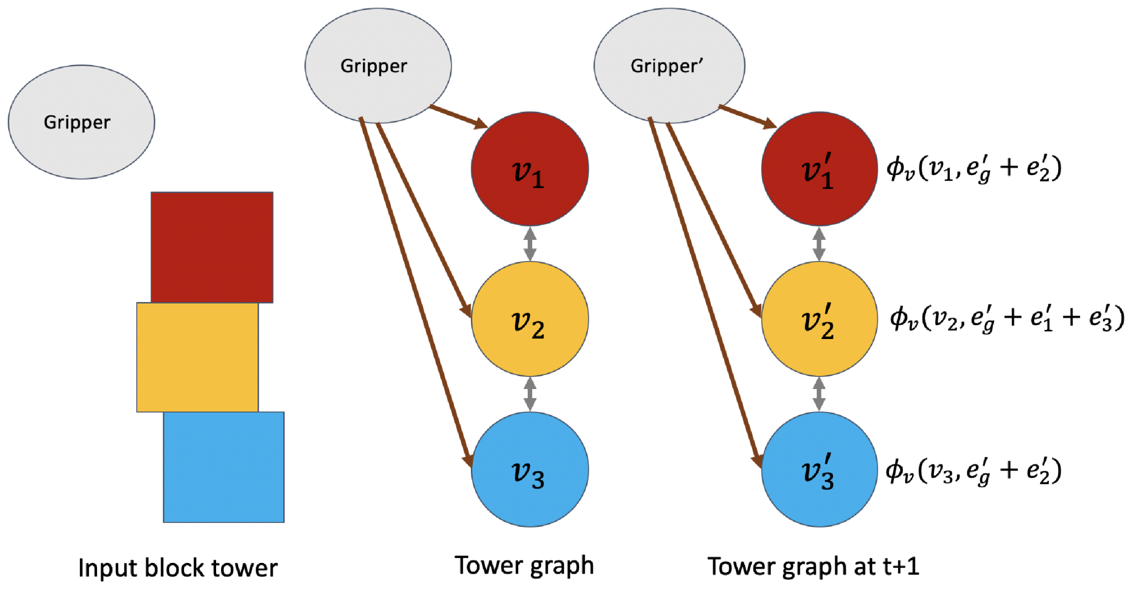


Figure 7-4: An example of how IMP could be applied to the moving tower problem shown in Figure 7-3. The tower is represented as a graph, as discussed in Chapter 3, with the gripper representing a separate action node. IMP is applied to learn the edge representation from the gripper, e'_g, and cluster it into different kinds of action *modes*.

problem as above – we must simultaneously learn an action embedding space, and a clustering of that space, which allows us to predict the future states of the blocks most accurately. Concretely, in a set of preliminary experiments, we applied Infinite Mixture Prototypes to jointly learn an edge embedding $e' = \phi_e(gripper, block_i)$ and a nonparametric clustering over e', optimized for the task loss of predicting the state of all blocks and the gripper at the next timestep (Figure 7-4). We found that IMP was very successful in discovering the action *modes* along with a representation of the scene which improved prediction. In the case of the 2D setup with pushing and picking affordances, IMP correctly discovered these two modes, and correctly classified 97% of actions into each type in a small test set, while simultaneously improving prediction performance by 5% over a method without modes. In the 3D setup, IMP successfully discovered 2 modes that corresponded to *stable* and *unstable* pushes, and again correctly classified between 90% and 95% of actions depending on initial seeds. It again improved in performance over a model without modes, reducing error from between 5 and 10% for next state prediction.

Extensions to this idea seem very promising for moving towards structure discovery in the space of learning to act. In future, this method could be applied to the domain of tool use, and would make specific predictions about tools being represented as a multi-modal set of clusters which depend on the *dynamics* of how the tool moves. Empirically, there is already evidence for this idea [101]; humans represent different grip points on a tool depending on whether those grips lead to distinctly different *dynamics* of how the tool moves. This result could be potentially predicted by the IMP for action model.

From a computational perspective, this method fits nicely into the literature on *skill learning* within robotics and planning. Most existing methods assume that demonstrations of skills for robots or agents are already segmented into "clean" demonstrations of each skill individually [240, 110]. IMP provides an opportunity for jointly segmenting and learning the skill dynamics, which better reflects the task that humans actually face.

7.3 Towards integration with motor planning

The kinds of tool use I have mostly discussed in this thesis has focused on *disembodied* tool use – placing objects in a 2D scene in order to cause the physical world to unfold in a particular way. Of course, people normally use tools in *embodied* settings as alluded to in Chapter 6. One of the reasons tool cognition is so fascinating is precisely because it combines low-level sensorimotor capabilities with high-level reasoning that is closer in spirit to language [208] and analogy [73, 24]. A critical direction moving forwards will therefore be to extend many of our computational models into the space of motor planning with tools (see Figure 7-5 for some examples across different animals, robots, and humans).

To do this, I believe the most successful framework will be Task and Motion Planning (TAMP) [116, 42, 218, 70]. In TAMP, problems are broken down into two levels: high-level reasoning, and low-level motor control. The central similarity across all TAMP methods is that the low-level motor movements are conditioned on the high-level actions which are chosen within a plan. For a thorough review of TAMP, please see [69].

In collaboration with Marc Toussaint, we developed a TAMP approach that represented this conditioning as the higher level of search providing *constraints* on the low-level continuous dynamics that would occur as a result of these higher level choices [218]. These

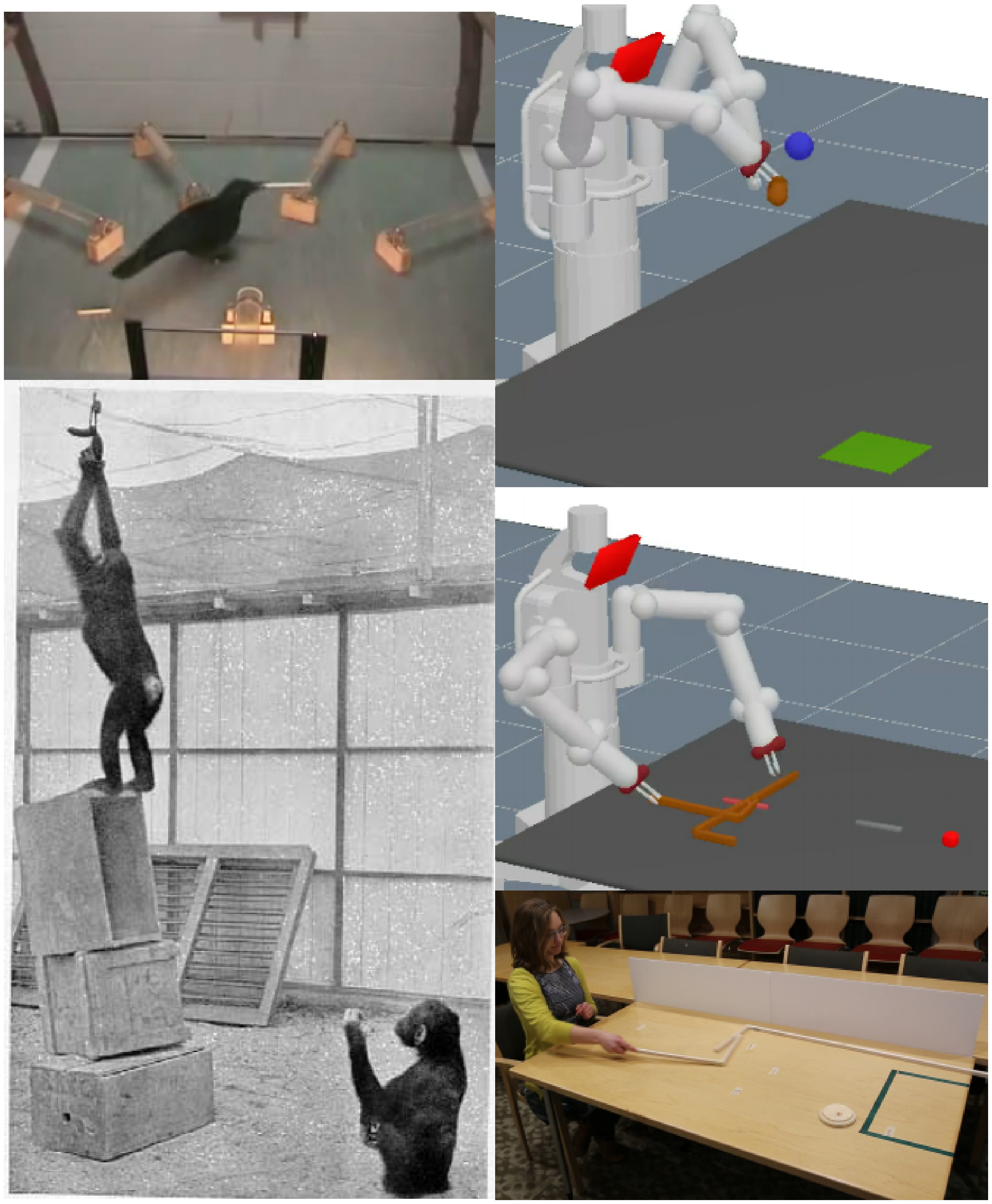


Figure 7-5: *Betty the Crow* (top-left) demonstrated the ability to use a sequence of hooks to retrieve a piece of food [233], and Koehler's apes stacked crates to reach a hanging bunch of bananas (bottom-left) [124]. Humans easily perform such sequential manipulation planning tasks naturally and flexibly (bottom-right). Figure credit: [218]

Table 7.1: Action operators and the path constraints they imply.

grasp(X Y)	[inside X Y] (staFree X Y)
handover(X Y Z)	[inside Z Y] (staFree Z Y)
place(X Y Z)	[above Y Z] (staOn Z Y)
throw(X Y)	(dynFree Y)
hit(X Y)	[touch X Y] [impulse X Y] (dynFree Y)
hitSlide(X Y Z)	[touch X Y] [impulse X Y] (above Y Z) (dynOn Y Z)
hitSlideSit(X Y Z)	“hitSlide(X Y Z)” “place(X Z)”
push(X, Y, Z)	komo(push X Y Z)

constraints could be either stable kinematic constraints, or *differentiable* dynamical and impulse exchange constraints at the path optimization level. By incorporating dynamical constraints, the method is able to solve a much wider variety of physical puzzles involving dynamic interactions, which previous methods were unable to successfully solve.

In the method, a set of geometric predicates and action operators are defined which describe the ways in which the robot can interact with the world. A set of these are shown in Table 7.1. Paired with each action operator are the path constraints that it implies on the low-level optimization. For details about the optimization procedure, including how we make the high-level search efficient, please see [218].

We applied this method to a set of problems inspired by human and animal tool use. First we showed that the model could handle these problems easily, and was additionally able to come up with multiple qualitatively distinct solutions for several of the presented problems. My main contribution to this paper was to then compare the solutions found by the model to those found by people for a subset of these same problems (Figure 7-6).

While this was a very preliminary study, the method shows some nice similarities in the sequence and frequency of the high-level actions used relative to our human participants. More interestingly, the differences between the model and human tool use execution suggest several areas for future work. The first striking difference is how often people “re-initiate” actions reactively, e.g. a frequent re-positioning of the tool to re-initiate pushing the same object. This underscores the importance of reactive and adaptive execution of plans in human motor planning relative to models that perform all planning “up front” like

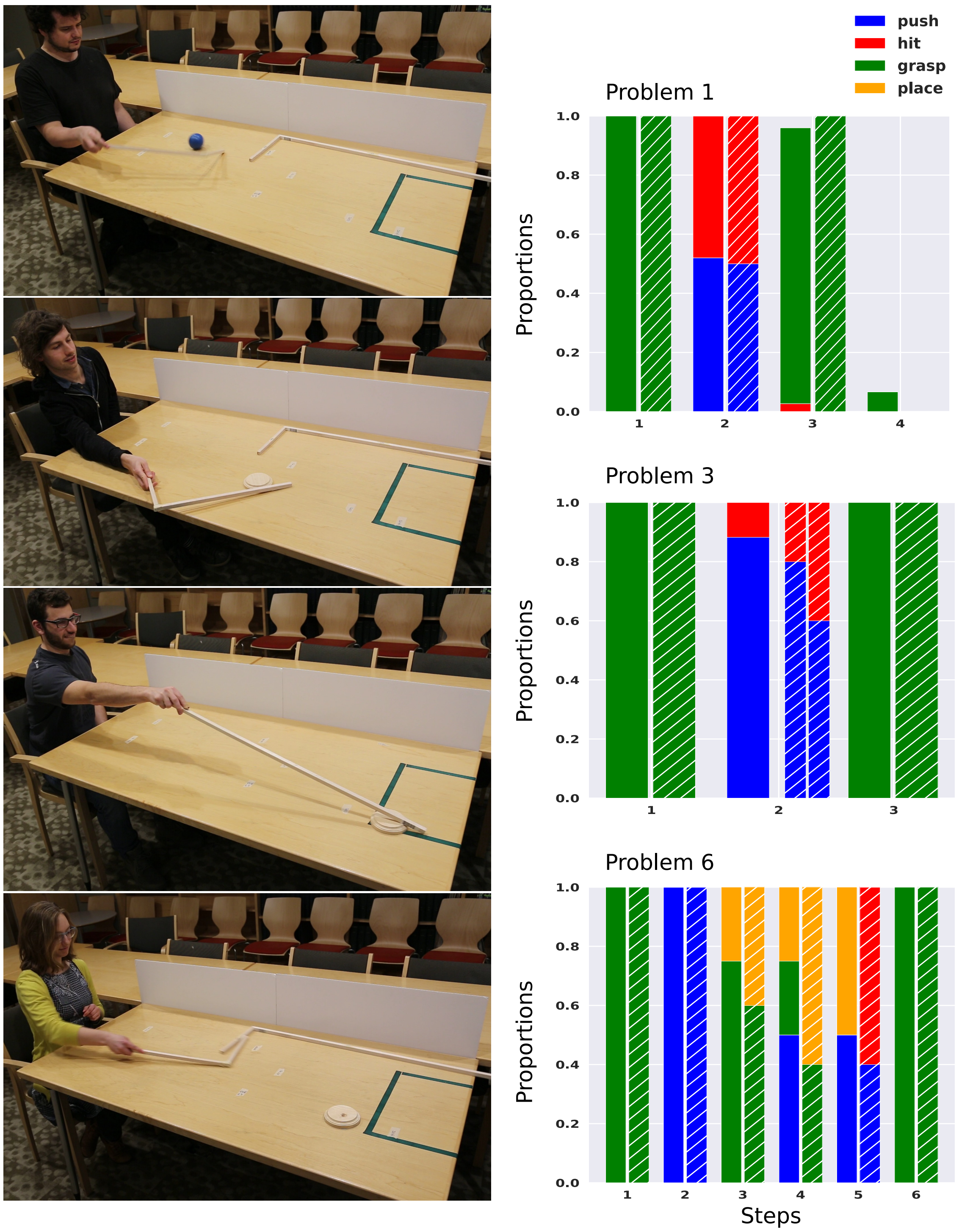


Figure 7-6: 5 subjects were studied on 4 tasks analogous to problems 1, 3, and 6 (where for problem 1 we tested also a high-friction puck replacing the ball). The right column displays frequencies of action primitives used along the high-level plan for our method (solid) and the humans (striped).

ours did. This was additionally a very important feature of human behavior in the Virtual Tools Game – action plans are frequently adapted if online monitoring shows that the current plan is not moving forward as expected. Second, the model presented here does not account for uncertainty or even knowledge of physical variables like mass or friction – these are *optimized* to allow the agent to solve the task. Human decisions clearly rely on perceptual estimates of those physical variables, as well as an understanding of their own motor noise [235].

Another exciting direction for future work is to focus on *learning* the modes that give rise to this rich, complex planning behavior. We hope that extensions of IMP mentioned in the previous section might be able to provide a mechanism for this. However questions remain about how to represent clustered representations in terms of *differentiable constraints* instead of next state prediction models.

Finally, we believe this approach to physics, namely interpreting it as a path optimization problem subject to particular constraints, might provide a unifying perspective for proponents of *qualitative physics* referenced in the introduction [147], and proponents of *simulation-based* physics [15] (which formed much of the basis of this thesis). If we interpret declarative expressions as constraints imposed by a high-level search procedure, then optimizing physics with respect to those decisions might often be computationally more straightforward than simulating everything in the scene. Constraints could, in effect, provide a mechanism for *attention*: which aspects of the simulation must hold, in which ways, for the resulting low-level path we observe to be possible? We believe this is fertile ground for future work, and have already begun setting up experiments to test this hypothesis with eye tracking.

7.4 The puzzling developmental trajectory of tool cognition and physics

A comprehensive body of research has shown that children improve in their tool using and tool innovation over the first several years of life [180]. By 24 months of age, children are sensitive to tool shapes and the functions those shapes afford, such as using a hook or rake object to acquire something out of reach [24]. It is not until children are at least 36 months of age that they can also choose appropriate tools in sequence to solve a multi-step tool-use problem, such as using a rake to obtain a stick, to then push a toy out of a tube [154]. In the space of manipulation, children develop over the course of 3-8 years of age in their ability to plan grasps that will result in an end-state that is comfortable [146].

By contrast, children's abilities to innovate or create tools out of available materials to solve these same tasks takes considerably longer to develop. In a set of striking studies Beck and colleagues [17] demonstrated that children were unable to create a hook shape from a pipe cleaner in order to obtain a toy within a tube until approximately 8 years of age, although they could easily use an already available hook shaped object to complete the task by age four.

There is considerable debate as to why these differences exist between children's tool innovation and tool using behaviors [180]. An increasingly popular explanation suggests that tool innovation is "ill-structured" relative to tool use [41]. The "shape of the solution" is not readily perceivable in tool innovation.

However, there are at least two alternative explanations. First, the effect of embodiment and real-world experience could be driving these differences. Perhaps developmentally, children have more opportunities to make and use tools as they age, and the simple sensorimotor experience of solving problems with objects drives the refinement of how children think about tools. Developmental results such as *functional fixedness*, which suggests that children become more regimented in using tools only in *typical* ways from 5 - 7 years of age [49], might support this hypothesis.

Second, differences between tool innovation and tool use could more simply be due to differences in the sizes of the search spaces children must navigate to find solutions to problems. In tool innovation, children must consider a very wide set of possible actions: any potential shape might be possible in order to accomplish a task. In tool use, the shapes have been given – children must only figure out the actions of which shape to use, and how to use it. *Learning to search*, especially through larger action spaces, may be one of the major roles which development plays in shaping children's tool cognition from inference

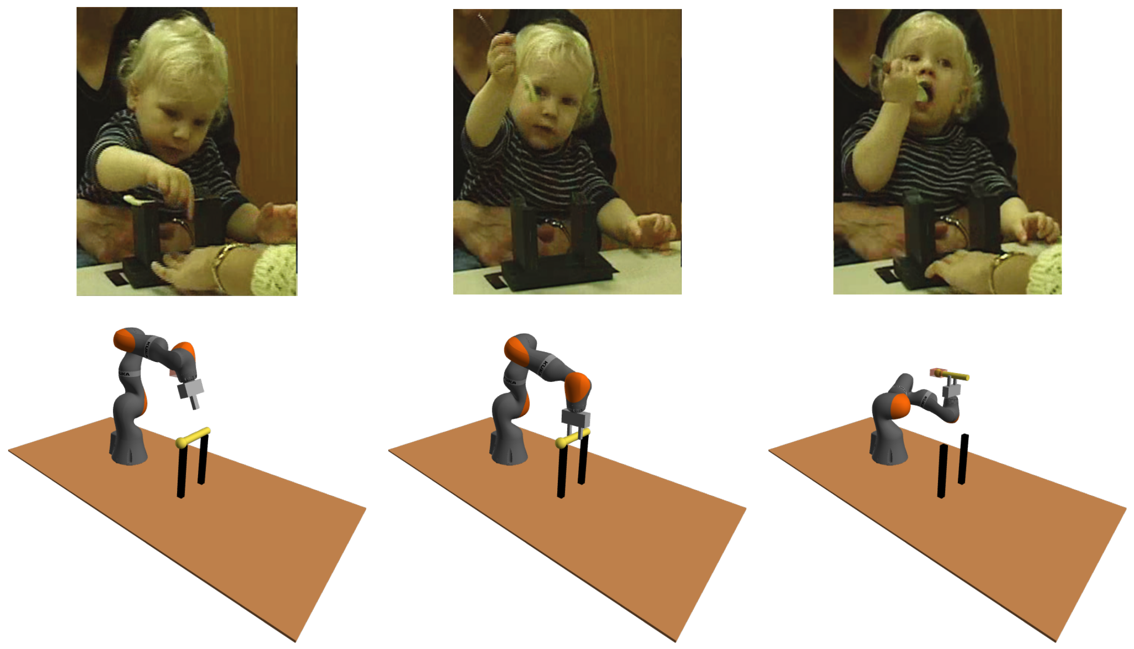


Figure 7-7: (a) a child performing the spoon task in study [146]—by choosing to grasp the spoon from above using their preferred hand, they finish the trajectory with an awkward grip (source: [119].) (b) task and motion planning robot using coarse geometric primitives executing a similar grip in our version of the spoon task. Figure credit: [138]

of tool function, to tool use, to tool innovation.

As a first step in investigating these hypotheses for the development of tool use, tool innovation, and motor problem-solving more generally, we have started to apply Task and Motion Planning methods to explain how children develop their tool using capabilities over the first several years of life (in collaboration with Joao Loula [138]).

Recall that TAMP suggests planning happens at two levels: the symbolic *task* space, and the low-level *motor* space. If that is the case, then we should expect the development of planning in children to be driven not only by their proficiency in simulating and executing any one given course of action, but also by their proficiency in searching this symbolic space for relevant actions in the first place.

In [138], we investigate whether such an explanation could account for two classic tool use tasks in children. The first task was taken from [146], where children between 9 and 19 months of age are presented with a spoon whose bowl is loaded with food, which they must bring to their mouth by grasping and manipulating the spoon. Critically, the spoon is initially placed on a two-column support, which allows the child to grasp it from the bottom or from the top.

The important manipulation involved simply changing the orientation of the spoon. When the handle is placed towards the child's dominant hand, a simple overhand grip will allow the child to successfully bring the food in the spoon to their mouth. When it is placed opposite to their hand, they would need to grip the spoon *from below* in order to avoid an awkward grasp to bring the food to their mouths (see the top part of figure 7-7 for an example of an awkward grasp on a difficult trial.). [146] found a clear separation between age groups for how they grasp the spoon: 9 month-olds tend to use the awkward grip, while 19 month-olds choose the efficient grasp from below. 14-month olds take a middle of the road strategy, and often change their grip halfway through.

We modeled this as a change to the level of abstraction over which they represent task-level plans. We propose that 9-month old children have a simplified representation of grasping an object, which depends only on the orientation of their hand, but not the orientation of the object, while 19-month old children also assume that grasping depends on the orientation of the object to be grasped. TAMP systems which differ only in this fidelity of representation of the *grasp* action predict exactly this difference in how children behave differently in these two groups. We suggest that part of development is therefore learning how to make abstractions over continuous states which allow for effective planning.

In the second study, we created a computational TAMP model to explain the results of [154]. [154] presented 36-month with two initial tasks, counterbalanced for order: one where they could retrieve an out of reach toy by pulling it with a rake, and another where they could retrieve a toy within a tube by pushing it out using a rod. These tasks were followed by a combination task, where once again the toy was inside a tube and could be pushed out with a rod, but now the rod was out of reach, and it was necessary for the child to first use the rake to retrieve the rod.

Though all children succeeded in both the rake and the rod task, only four out of sixteen of them managed to solve the combination task without assistance. After being given verbal hints to the solution, however—the strongest of which is "You can use this (the experimenter taps the head of the rake) to get this (tapping the rod) and then get the toy."—most children succeeded at the task.

We modeled this experiment using TAMP over an appropriate set of symbols and modes, and showed that the difference in search complexity was substantial. In the case of either the rake or the rod task, the number of nodes in the search tree that needed to be expanded was only 3, but in combination this explodes to 105. With a hint, that drops to 1. We therefore suggest a simple interpretation of the results in [154]: even though children are likely to reason symbolically about tool-use and use that reasoning to successfully guide their low-level plans when retrieving a toy with a rake or a rod, the apparent simplicity of a plan that requires chaining these two actions together can actually render the problem intractable for some types of planners by blowing up the search space.

Under this view, the problem is not with performing actions not directly related to the goal, which children successfully do in this and other tasks, but in discovering procedures to more efficiently navigate such large search spaces. The blow-up we observe in the combination task stems mostly from the choice of primitives and the naive breadth-first search procedure for the task-level plan—it is likely that, as children grow older, they develop sophisticated abstractions and heuristics for planning, avoiding such tractability problems. For instance, one could imagine that after solving the first two tasks, a child could learn to represent each of them as a single, more abstract predicate—in that case, the combination task would be significantly easier, requiring only two rather than six predicates to solve.

In future work, we hope to expand on these findings by explaining more of the developmental tool use literature: exactly how are children's abstractions for actions developing over the first few years of life? And how does that interact with their changing knowledge of physics? We hope such investigations will lead to both a new way of thinking about physical prediction, and also clear up many of the mysteries of why children can have such rich physical knowledge without being able to always use it successfully to take actions.

7.5 Beyond using tools: making tools and cumulative technological culture

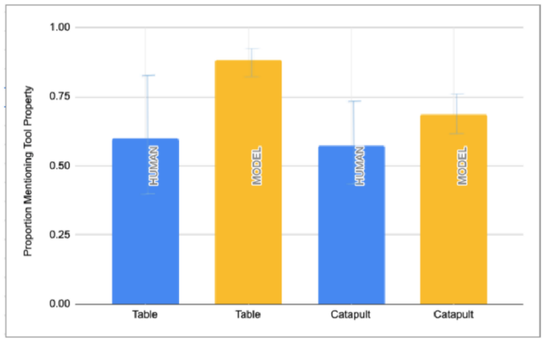
Although this thesis almost exclusively focused on how people use unfamiliar tools to solve novel problems, the Virtual Tools domain opens up many new avenues to build computational accounts of how people innovate new tools, modify existing tools for new purposes, and even pass on their tool knowledge to others to enable the creation of even better tools in the future.

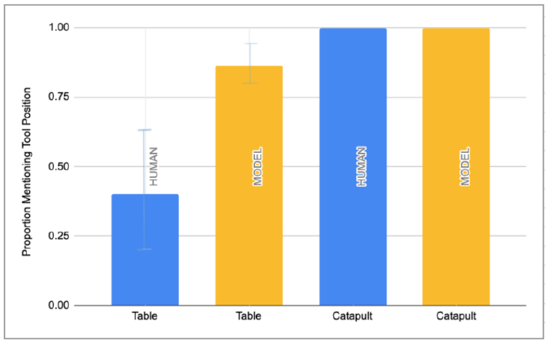
Cumulative technological culture (CTC) is one reason why so many scientists are excited about tools. Finding evidence of cumulative technological culture in other species has been a great quest for the animal cognition community given its rarity, with only a few known species of primates and birds demonstrating all components of cumulative culture in the wild [184, 153, 188].

It has recently been proposed that CTC is mainly supported by a singularly impressive human ability: mechanical reasoning [168] (which we refer to as intuitive physics throughout this thesis). But mechanical reasoning alone is clearly not sufficient for the efficient transfer of knowledge between individuals. So the big question remains: why and how do people communicate so effectively about physical reasoning?

[174] suggests that even young children are excellent at determining whether an artifact was either socially transmitted through copying, or designed from scratch for a particular problem. [152, 214] have similarly shown that people can cumulatively iterate on the design of an arrowhead to make it particularly effective for a given task. But the Virtual Tools game allows us to explore these questions in a much richer space. Could people learn and pass on a "catapulting" strategy? To do so, do they correctly pull out the functionally relevant features of tools and scenes for catapulting – like using a heavy tool, and ensure that there's actually a catapult in the scene to interact with? When is a demonstration of an action sufficient vs. providing a linguistic explanation?

In preliminary experiments, we have some initial data to begin looking at some of these questions. We asked participants to play the same set of levels that we used in Chapter 5 to look at strategy learning. However, we also asked them to provide a hint at the conclusion





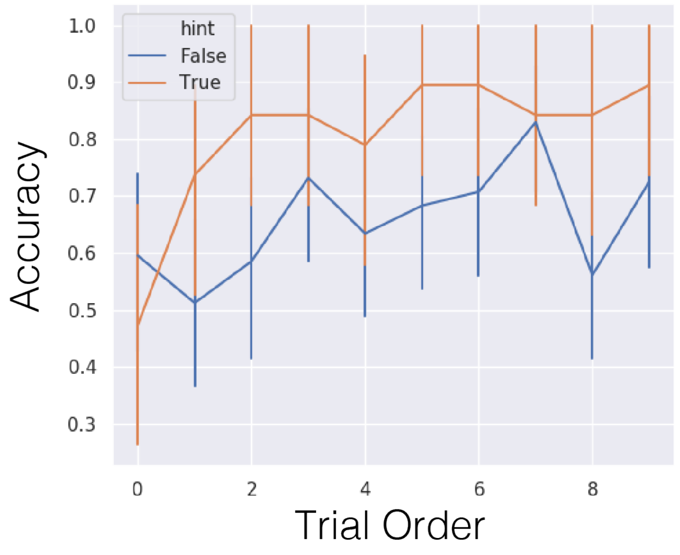


Figure 7-8: Top Left: Proportion of hints featuring information about where a tool should be placed. Top right: Proportion of hints featuring information about tool properties (such as height, size, etc.). Bottom: Improvements across training when given the hint (orange) vs. without a hint (blue).

of the experiment that would help someone else playing the game. We then gave those hints to new players to examine how much they helped participants relative to controls who were not given a hint. While we have very limited data, we see promising trends in people using terminology in their hints reflecting the kinds of strategies discovered by the Relational Program Policy (RPP) model from Chapter 5, and that these hints did help new participants solve the levels faster relative to controls (Figure 7-8).

We then looked at how a *single demonstration* of a strategy might affect how people solve new levels that require the same strategy, but look visually distinct (Figure 7-9). Again, while the results are preliminary, what we see is very promising: a much greater number of participants take actions (from the very first attempt) that are aligned with the

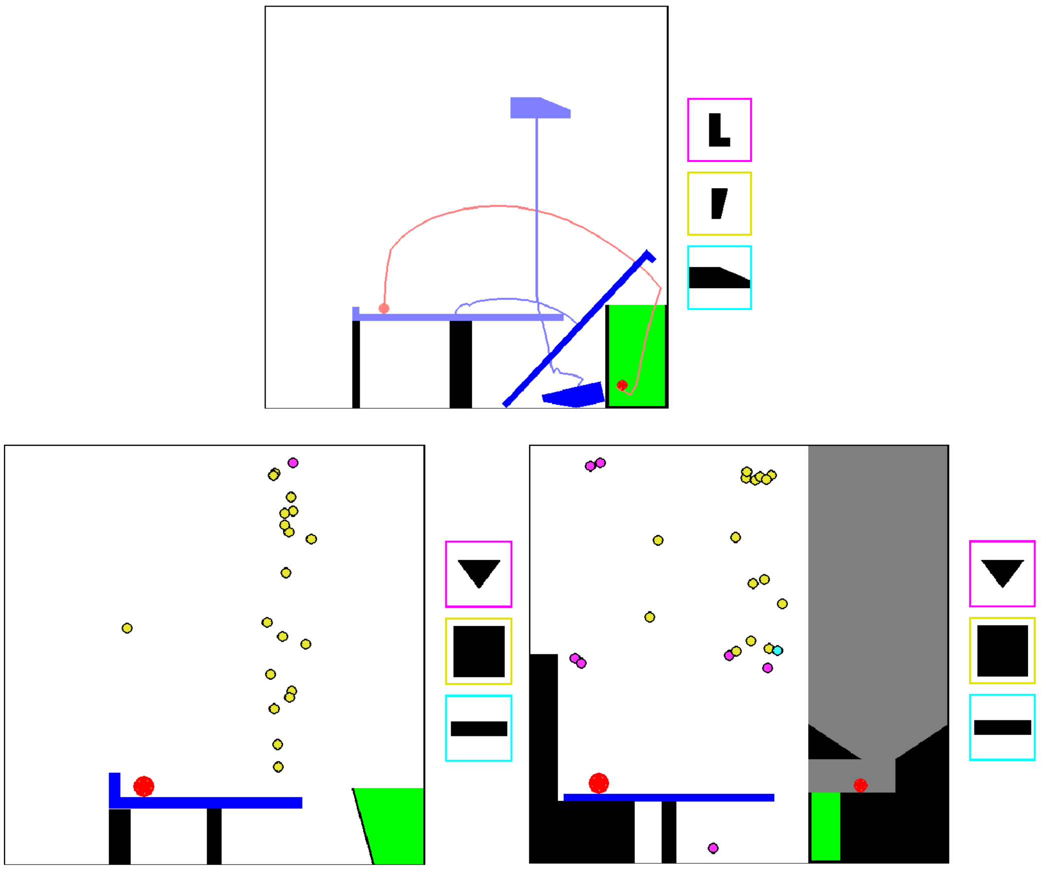


Figure 7-9: Top: Demonstration of a catapulting strategy. Bottom: First placements for test catapulting levels that use different tools, sizes of balls, and potentially look visually distinct from the observed level.

correct strategy. Figure 7-9 shows this for the case of catapulting, where many participants pick up on the size of the tool as being important, and correctly try to catapult the ball into the goal, relative to untrained participants.

We are excited to build on these results to better understand the interaction between learning, communication, and the *cultural ratchet*. The Virtual Tools domain provides a new opportunity to bring a microscope to the problem of communication for action. How do people communicate effectively to create, modify, and use new tools? How do new capabilities offered by new tools then support a *cultural ratchet* of increasingly complex artifacts and skills? Future work will tell!

Chapter 8

Conclusions

Humans have an unparalleled ability to solve new problems efficiently by relying on sophisticated mental models and structured search spaces in which to explore them. Mental simulation alone is not sufficient, nor are structured action spaces. Instead, both must be used in concert to achieve both *flexibility* and *efficiency* in physical problem-solving.

This has important implications for both computational cognitive science and AI. Cognitive scientists must embrace more complicated settings for action and the complexities of how it interacts with perception and simulation. By examining representations not just through the lens of inference, but through the lens of action, cognitive science may arrive at far different conclusions about how people represent the world. The "Sample, Simulate, Update" and "Relational program policies" models are only first steps in this direction; many of the most interesting, unanswered questions lie at the intersection of sampling, simulating and updating. How might world models be represented to allow more efficient updating? How might sampling mechanisms be structured to reflect the world models in which they are learned? Task and Motion Planning may be a good computational framework for addressing these questions in the years to come.

While model-based methods are gaining traction in machine learning [189], there are still limitations to overcome in both their asymptotic performance and how to represent models in ways that support action [157, 30]. Part of this thesis emphasizes that models are *not enough* – how an agent uses the model, and the synergistic interaction between the specification of the model and the specification of the action space is perhaps more important. To create machines that are as flexible as people, they will need to be more structured – whether that comes from discovering the structure in a data-driven way, or whether that structure is imbued by a programmer in a sufficiently task-general manner. In either case, machines, like people, must be able to *rapidly* adapt to *new structures* in actions and problems that will afford generalization to problems that could not have been foreseen.

This thesis provides a path for joint work in the space between AI, robotics and cognitive science. By focusing on physical problem-solving, we present a substrate for each community to make contributions towards a shared picture of flexible, efficient intelligence. The physical problem solving toolbox has been made, we must continue to fill it with tools.

Appendix A

Supplement for Rapid Physical Problem-Solving with Mental Simulation

A.1 Experiment

A.1.1 Procedure

Participants were recruited from Amazon Mechanical Turk using the psiTurk framework [93]. We recruited 94 participants for the first experiment and compensated them \$2.50 for 15-20 minutes of work. For the validation experiment we recruited 50 additional unique participants and compensated them \$2.00 for slightly less than 15 minutes of work.

On each trial, participants were initially presented with a freeze-frame of the scene and a goal description (physics switched off) (Fig. 4-1D(i)). They were instructed that all of the black objects were immovable, but when physics was turned on, the blue and red objects may move. They were provided with three 'tools' that they could place anywhere in the scene (that did not overlap with other objects, goals, or out-of-bounds areas), by clicking on a tool and then clicking where they would like to place it (Fig. 4-1D(ii)). This object would then be added as-is to the scene; participants could not rotate or rescale it. As soon as this tool was placed, physics would be switched on using a Javascript version of the

Chipmunk 2D physics engine [72, 130], and all movable objects would start to fall under the force of gravity (Fig. 4-1D(iii)); after this participants could no longer intervene on the scene. However, participants could click a 'reset' button at any time to return the scene to its initial state and try another action. Participants were given two minutes to solve the problem – if they solved it they could move onto the next level immediately. Otherwise, they could choose to move on any time after two minutes had passed. Within each trial, we recorded all attempted actions: which tool was used, where it was placed, and the clock time when it was placed since the start of the trial. See `https://sites.google.com/view/virtualtoolsgame` for videos demonstrating this procedure.

To familiarize participants with the experiment, we initially instructed them on the way the game worked, then gave them a 'playground' level with no goal to introduce them to the dynamics of the world. Participants had to remain in the playground for at least 30s and try at least two tool placements before moving on. Finally, participants were given two simple practice levels that they were required to solve before the main part of the experiment began; these were not analyzed.

Participants were asked to solve 14 of the 20 levels: all eight unpaired levels and one each of the six paired levels (so that learning in one instance of a pair would not affect performance in the other). The choice of which instance of a pair was determined randomly at the start of the experiment. The order in which levels were presented was additionally randomized.

In the validation experiment, all participants were asked to solve all 10 levels. Other than the level choice, all procedures and introductory materials were identical.

A.1.2 Data cleaning

To standardize results across participants, we only analyzed the first 120s of play, even though participants were allowed to continue play past the 120s mark. In the first experiment, this affected 7.1% of all trials (93); of those, 33 were eventually solved. In the validation experiment, this affected 16.8% of all trials (84), of which 25 were later solved. To further standardize results between participants and the model, we treated any actions that would have accomplished the goal within 20s as successes, regardless of whether participants reset the scene before the success could be counted or waited longer than 20s for a solution; this caused 4.5% (59) of all trials to be analyzed differently than participants experienced them in the first experiment, and 3.2% (16) of all trials in the validation experiment.

A.1.3 Learning over the experiment

While participants improved in solution rate over the course of the first experiment (76% solution rate on the first three trials to 86% on the last three; $\chi^2(1) = 9.7$, $p = 0.002$), they did not solve the levels more efficiently ($\chi^2(1) = 2.0$, $p = 0.15$), taking an equal number of attempts to arrive at a solution across the experiment.

A.2 SSUP model implementation

Algorithm 1 demonstrates how the SSUP model is implemented to perform the Virtual Tools game. Further details are below.

A.2.1 Implementing *sampling*: an object-oriented prior

The object-oriented prior can be described as a categorical sample on which object to interact with, and a Gaussian distribution specifying the position of the tool relative to that object. Formally, the generative procedure for the prior follows:

- Sample a dynamic object in the scene $object \sim Multinomial(\{\frac{1}{n_{obj}} \| i \in [0, ..., n_{obj}]\})$
- Sample a position y relative to that object, using a Gaussian distribution parameterized with mean $object_y$ and standard deviation σ_y, truncated on each side by the legal lowest and highest positions respectively
- Sample a position x relative to that object:
    - Compute the left and right edges of the bounding box for that object, BB_{left} and BB_{right}

Algorithm 1: SSUP model for the Virtual Tools game

```
Sample $n_{init}$ points from prior $\pi(s)$ for each tool
Simulate actions to get noisy rewards $\hat{r}$ using internal model
Initialize policy parameters $\theta$ using policy gradient on initial points
while not successful do
    Set $acting = False$
    With probability $\varepsilon$, sample action $a$ from prior
    With probability $1 - \varepsilon$, sample action $a$ from policy
    Estimate noisy reward $\hat{r}$ from internal model on action $a$
    if $r > T$ then
        Set $acting = True$
        Try action $a$ in environment
    else if $i \geq n_{iters}$ then
        Set $acting = True$
        Try best action $a^*$ simulated so far which has not yet been tried
    if acting then
        Observe $r$ from environment on action $a$.
        If successful, exit.
        Simulate $\hat{r}$ assuming other two tool choices.
        Update policy based on all three estimates and actions.
    else
        Update policy using policy gradient
```

- Sample a value v uniformly between $BB_{left} - \sigma_x$ and $BB_{right} - \sigma_x$.
- If $v < BB_{left}$ or $v > BB_{right}$, sample x from a normal centered on the edge of the bounding box with standard deviation σ_x.
- Otherwise, $x = v$.

This has the effect of sampling mostly uniformly around object extents in the x direction, but otherwise dropping off around the edges of objects proportionally to σ_x.

σ_x and σ_y are then free parameters which were chosen to reflect a relatively uniform prior in y and a tighter distribution in x. These decisions were examined using a sensitivity analysis shown in Figure A-1.

We experimented with an alternative geometric prior, which could sample "above", "below", "left", "right", and "middle" of objects before then committing to Gaussian positions respecting that geometric decision, but found that this did not perform better than the more uninformed prior. We therefore use the more uninformed prior to reduce the number of free parameters in the model.

To initialize search, the model first samples a number of initial points, $n_{initial}$, for each tool from this prior, and runs each action through the noisy simulator. The policy is initialized with these noisy reward estimates. Not initializing the policy in this way is detrimental, as can be seen in the sensitivity analysis.

A.2.2 Implementing *simulation*

The noisy simulation engine within the SSUP model is meant to capture the essence of the human Intuitive Physics Engine [15]. It is based on the Chipmunk physics engine just as the experiment is, but introduces stochasticity in the dynamics when objects collide, similar to how the uncertainty in human physical predictions increases when they must simulate through collisions [202, 98]. Collisions are made stochastic by inserting noise into the direction that objects bounce off of each other, and how much energy is transferred. This is accomplished for each collision by adjusting the direction that collision forces are applied (ϕ) and the elasticity (bounciness) of the collision (e), by adding noise according to two parameters (σ_ϕ, σ_e):

$$\begin{aligned} \phi' &\sim wrappedNormal(\phi, \sigma_\phi) \\ e' &\sim \mathcal{N}(e, \sigma_e) \; s.t. \; e' > 0 \end{aligned} \tag{A.1}$$

The SSUP model runs n_{sims} simulations (set here at 4) per imagined action to determine an average reward (r; see Section A.2.3). This reward is then used to update the action-outcome policy parameters θ, and to decide on whether to take an action.

A.2.3 Implementing *updating*

The main body of the SSUP model is the simulation-action loop, which updates the policy π with reward estimates from sampled actions. The policy π consists of first choosing which tool to use, and then conditioned on each tool, where to put it in the scene. We model this as a mixture of Gaussians, one Gaussian for the position of each tool. This gives 2 parameters for the tool weights because their probabilities have to sum up to 1, and 4 parameters for the Gaussian position and variance on each tool (for a total of 2+12 parameters). In principle, the SSUP framework is agnostic to the particular form of the update. In practice, we found that a simple policy gradient worked well, outlined below, but other methods such as Bayesian optimization resulted in similar trends amongst the tested ablations and full model.

We use a simple policy gradient algorithm [231] to update our policy parameters:

$$\theta \leftarrow \theta + \alpha r \nabla_\theta \log \pi_\theta(a) \tag{A.2}$$

where a is the action taken, α is the learning rate, r is the reward, and θ are the policy parameters to be estimated.

The policy is initialized such that each Gaussian is centered in the middle of the screen with isotropic variance of 50px (1/12th of the height and width). It is then updated using noisy reward estimates from n_{init} sampled actions from the prior before any actions are taken. We found this dramatically improved stability and performed much better than a policy which was parameterized with respect to the objects in the scene.

We include exploration in our action selection, using an epsilon greedy strategy. ε is considered to be a free parameter of the model. When exploring, we sample actions from our object-oriented prior, outlined in section A.2.1.

SSUP continues to sample actions until it either finds an action with a reward greater than a given threshold, T, or until it reaches a maximum number of simulations, n_{iters}. If it finds an action with high reward, it executes that action in the environment. Otherwise, if it reaches the maximum number of internal simulations, n_{iters}, it takes the best action it has simulated so far which has not been tried in the real world. If it succeeds, the model is finished. Otherwise it resumes simulation.

Counterfactual updates

Whenever our model takes an action in the environment, it additionally queries its noisy simulator for what would have happened if it had used the other two tools. The policy gradient is therefore calculated on three data points: one from each of the possible tools. These are also counted as the first two simulations of each inner loop. We found that this stabilized the policy gradient, such that it could determine whether the reward was due to the tool used, or the position chosen. We imagine that such an update might be generally useful in increasingly structured action spaces when a strong model is available.

Reward function definition

Our reward is defined as the normalized minimum distance to the goal along the observed or simulated trajectory. This reward function was provided for every model that we considered. Normalization is performed with respect to what the outcome of this metric would be if the agent had taken no action, such that the reward can be calculated as:

$$r = 1 - \frac{min_{t=0,N;obj \in objects} d(obj, goal, action)}{min_{t=0,N;obj \in objects} d(obj, goal)} \tag{A.3}$$

where d is the distance between a goal object (red object) and the goal, under a particular action; N is the total number of time-steps in the trajectory. The subtraction is to flip signs such that getting into the goal (at a distance of 0) results in the highest possible reward.

This is therefore a measure of *intervention* rather than generic distance to the goal. Across all experiments, we found this reward function to perform substantially better than one which was based only on the unnormalized minimum distance metric.

A.2.4 Running on the Virtual Tools game

The SSUP algorithm described in Algorithm 1 will continue to run until it finds a solution. However, to match human performance under time limits, we capped the number of actions that the model could take to 10, which was the median number of actions people took on trials for which they did not solve the level. If the model did not find a solution within 10 actions, it was considered to be unsolved.

A.3 Physical parameter tuning model details

As an alternate learning scheme, we suppose that people use observations from failed actions to refine their internal dynamics models for the game, and then use those updated models to choose the next action. Critically, this alternate learning scheme *does not* sample actions from a policy which is updated from observations. It always samples actions from the object-centered prior, and only the noisy dynamics model can be improved and refined based on observations.

We instantiate this learning scheme using a model that tunes parameters within its physics engines that are related to object dynamics: object densities d, frictions f and elasticities e, which we collectively call ψ. This model relies on the same physics engine as the full SSUP framework, including uncertainty introduced during collisions (see Section A.2.2).

After each failed action is performed, the parameter tuning model attempts to update its internal parameters to match the observed outcome of that action using Bayesian inference; this is an instance of "analysis-by-synthesis" [121]. The model observes 20s of a failed trajectory, and samples the location of all movable objects at 50 points evenly spaced in time. To approximate the likelihood of each observation given a parameterization ψ, the model produces 20 simulations of the same action, and records the positions of all movable objects in that simulation at the same time points. From these records, the model takes the positions of each object at each time point, and parameterizes a Gaussian over its likelihood of where that object should be at that point in time $(\mu_{obj}^t, \sigma_{obj}^t)$. The likelihood of observing the full trajectory is therefore the product of each of the individual observed object positions at each time point:

$$P(O|\psi) = \prod_{obj}\prod_{t} P(o_{obj}^t|\mu_{obj}^t, \sigma_{obj}^t) \tag{A.4}$$

We parameterize the prior for each property as a Gaussian centered at the true value for that property (1 for density, 0.5 for elasticities and frictions), with variance 0.025. Our proposal function is defined as a set of truncated Gaussians with variances of 0.02 (and mean equal to the current estimate).

We implement this inference using the Metropolis-Hastings (MH) algorithm for 20 iterations using the likelihood function and proposal above, using 5 chains and 5 burn-in samples. We found that this was enough to give reliable estimates for each of the parameters. This procedure was implemented in the probabilistic programming language Gen [39].

Every time a new action is observed, we initialize the MH procedure using the parameters determined from the previous observation. In this way, learning (in the form of more refined priors) can occur throughout a set of actions within a level.

A.4 Parameter sensitivity analysis

To ensure that the choice of parameters did not unduly affect model performance, we tested how well the SSUP model performed under different settings of each of its parameters. For a measure of model performance, we used a metric called total Area Between Cumulative Solutions (ABCS). This metric is defined for each level as the area between the human and model cumulative solution curves (see Fig. 4-6A in the main text), normalized between 0 (perfect match) to 1 (participants or the model always solve the level instantaneously, while the other never solves the level). This metric combines the by-trial accuracy, number of actions used, and evolution of solution rate into a single metric.

Because the SSUP model includes 10 parameters, we could not reasonably test model performance across a full grid of parameter choices; instead, we test model performance along a single dimension at a time, varying one parameter but keeping all others constant. As can be seen in Fig. A-1, model performance did not differ significantly across a wide range of parameter settings around the values used in the SSUP model. This suggests that more precise but computationally intractable parameter fitting would lead to at best marginal improvements in model fit.

As further evidence that these parameters generalize, we used an identical parameteri-

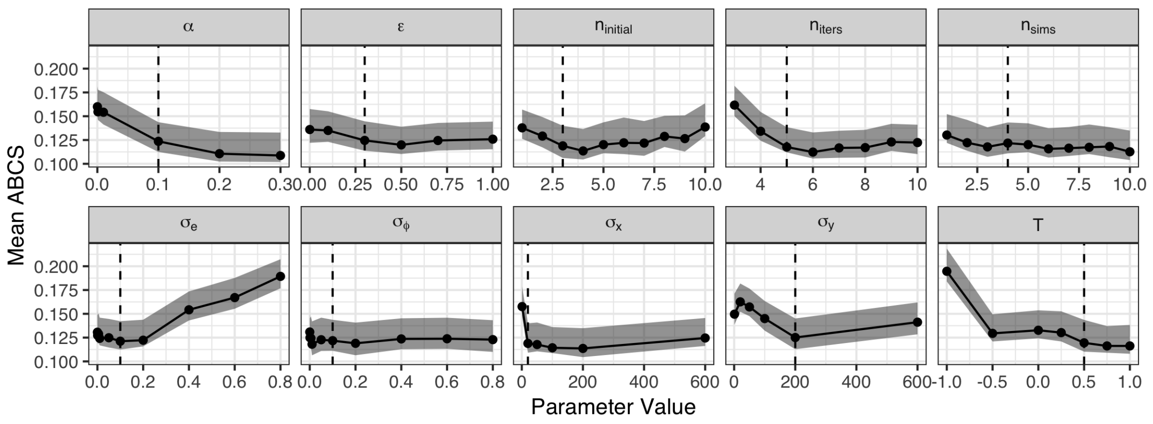


Figure A-1: Mean Area Between Cumulative Solutions (ABCS) values across all trials for different parameter values, keeping all other parameters at the default values. The default values are denoted by the dashed lines. Grey areas indicate 95% bootstrapped confidence intervals. Legend: α: learning rate for policy gradient, ε: the probability of sampling an exploratory action from the prior, $n_{initial}$: number of initial samples from the prior used to initialize the policy, n_{iters}: number of internal iterations before action must be taken, n_{sims}: the number of simulations run for each imagined action, σ_e: the noise (in radians) for collision elasticity, σ_ϕ: the noise (in radians) for collision direction, σ_x: the standard deviation of the prior in the x direction, σ_y: the standard deviation of the prior in the y direction, T: -1$\times$ the reward threshold for acting

zation of the model for the 10 validation levels, and found good fits to human data there as well.

A.5 A Deep Reinforcement Learning baseline

We looked at whether a popular approach from deep reinforcement learning, Deep Q Networks [156], could solve the Virtual Tools game from pixel inputs alone.[1] While we expected that such approaches would require significant amounts of experience to learn useful representations, we hoped it might discover generally useful priors (like being object-oriented, or even understanding whether something should be placed above or below another object) that may transfer across different level types.

[1]We also considered Proximal Policy Optimization [190], but were unable to train the network to perform above chance levels, even within a single level template; see [3].

A.5.1 Random level generation

For the purposes of comparing human and model performance in the first experiment, we used 20 hand-designed levels. However, Deep Q Learning requires large amounts of data for training, and though we could allow training by taking extensive actions on the given levels, this would risk over-learning particulars of those 20.

Instead, we randomly generated levels from a set of five templates based on five of the hand-designed levels. These templates were designed such that they contained the same set of objects, and could be solved in similar ways, but the sizes and configurations of objects could vary, which enforced some variability in the solution actions. The specific algorithm for generating levels varied by template, but involved resizing or shifting many of the object, subject to some geometric constraints (e.g., the 'tabletop' in each Table level was always between the slope and goal, and the ball in each Catapult level always rested on the catapult object). See Fig. A-2A for example scenes from each template.

Each generated level included three tools randomly drawn from a pool of possible shapes that were used to construct tools for the 20 hand-designed levels. These tools could further be randomly resized or rotated at angles of 90 degrees, subject to the constraint that they continued to fit in the 90×90 pixel area that each of the original tools fit into. See Fig. A-2B for examples of randomly generated tools.

To guarantee that each level has a reasonable but non-trivial solution, we proposed a "solution region" for each template that comprised a similar area to the primary solution for the base level. For instance, the solution region for the Basic template was a rectangle above the key ball, while the solution region for the Shafts template contained both areas directly above the shafts. We then randomly generated 100 tool placements from this region and selected only trials for which between 3% and 80% of actions would solve the level.

We used these templates to generate 1,000 levels each. These levels were then split to provide 900 training levels for each template, and 100 validation levels.

A.5.2 Architecture details

Our DQN architecture takes in an image of the screen and images of each tool in order to compute Q values for a discretized version of the space. The images are down-sampled such that the screen is 90x90 pixels and each tool is 30×30 pixels. We choose a discretization of 20×20 which is sufficient to solve most of the levels, and found this reliably converged to a reasonable policy after 150,000 iterations.

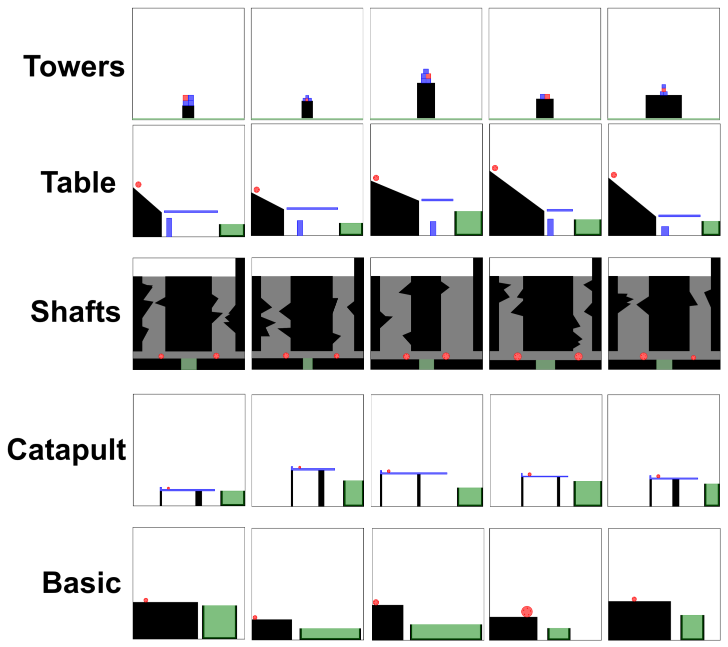
The architecture consists of four networks: a tool image network, a screen image network, a tool policy network, and a position policy network. We run each tool image through the network, fusing this with the screen image run through the network before passing the fused representation to both the tool policy and position policy networks. This gives us a 400-dimensional action vector for the position, and a 3 dimensional action vector for the tool choice.

- The tool image network consists of 4 convolutional layers, with 12, 24, 12, and 3 channels respectively at each layer. The kernel sizes are 3, 4, 3, and 2 respectively.
- The screen image network also consists of 4 convolutional layers, with 32, 64, 32 and 16 channels respectively, and kernel sizes 8, 4, 3, and 2, following the original DQN Atari network [156].
- The tool policy network is a simple 2-layer multi-layer perceptron (MLP) with 100 hidden units.
- The position policy network is similarly a simple 2-layer MLP with 100 hidden units.

We use epsilon greedy exploration during training, with a linearly decreasing epsilon schedule. We use a batch size of 32 and a constant learning rate of 2.5×10^{-4}. For optimization, we use RMSProp [106].

A.5.3 Training and results

We considered two training schemes for DQN: training on only a single level template, or training on levels from all templates at once. Training continued until the models observed 150,000 instances, at which point performance appeared to stabilize for all models



(a) Randomly generated level screens

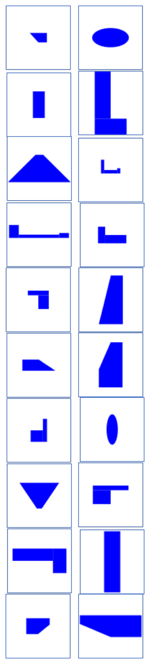

(b) Randomly generated tools that are paired with level screens

Figure A-2: Randomly generated levels for deep learning baselines

(see Fig. A-3A). The reward function for DQN is identical to the one used for SSUP: a normalized reward with respect to what would have happened if no action was taken.

Learning curves for each model are shown in Figure A-3A. In all cases, models trained specifically for one level solve the validation levels from that template more often than the model trained across all levels (Fig. A-3B). Indeed, the model trained on all levels seems to have learned a "launching" strategy which is sometimes successful for those levels which involve launching an object (Basic, Catapult, Shafts and Towers). However, it is not able to simultaneously learn a "supporting" strategy for the Table template, where solutions require supporting an object from below.

These training results suggest why the DQN + Updating model fared so poorly in comparison to the SSUP model: while it is possible to learn policies from action-reward feedback in a single scenario, it is difficult to do so for multiple scenarios simultaneously. Thus, even when we include an update mechanism to support rapid learning within a trial, the model-free policy learned from extensive experience does not guide the updating ap-

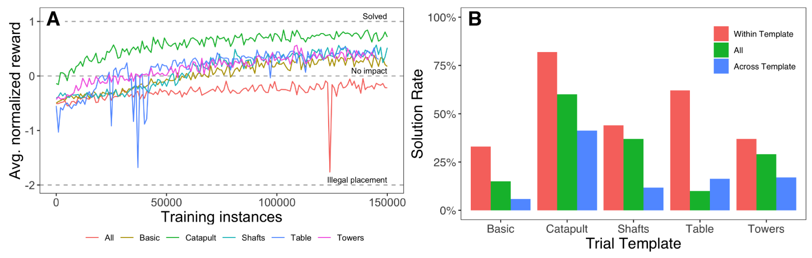


Figure A-3: **(A)** Training curves for DQN training on individual level templates, or all templates combined (All, red curve). Reward is averaged over each epoch of 1,000 training instances. Models trained on individual scene templates are able to consistently choose successful or promising actions; however, the model trained to generalize across templates under performs, not even consistently achieving a reward that is better than having no impact on the scene. Note that this is because the model has over-specified to some levels and not others. In some levels it has positive impact, and in other levels it has negative impact. **(B)** Accuracy of DQN on the validation template levels. Models are grouped by training regime: within template, all templates simultaneously, or averaged across all models trained on alternate templates.

propriately across the variety of levels used in the Virtual Tools game.

A.5.4 Future extensions

There is exciting work currently being developed to set up multi-task reinforcement learning benchmarks, such as OpenAI's Sonic and Coin Run challenges [162, 32]. The Virtual Tools game expands on this growing set by providing a strong test of transfer learning for physical planning and acting, but additionally requires few-shot learning within the test domains.

While we see the Virtual Tools game as presented to be a challenge for current AI approaches, we also consider ways of extending it to keep pace as a challenge problem for future AI. We could make the action and state space more complex by requiring sequential tool placements. Or, inspired by the way that crows and children can shape objects to make tools [197, 17], we could require agents to design their own tools instead of selecting from a set of provided objects. We could also make the dynamics more complex by introducing objects with various densities, frictions, and elasticities, which would need to be learned through interaction. Each of these changes would be a simple extension to the game, yet we would expect them to push the boundaries of artificial agents.

A.6 Additional results

A.6.1 Analyzing differences in behavior for matched trials

We used a similar mechanism to evaluate whether there was any difference in actions taken between matched level pairs (A and B). Specifically, for each of level A and B, we use a leave-one-out cross-validation procedure to obtain the probability that each participant's action came from the level they actually played (A or B) as opposed to the alternative matched level (B or A). We fit a Kernel Density Estimate (KDE) plus background guessing to the actions taken by all the people in the opposing level, as well as all actions except the participant's in the true level. We can then obtain likelihood estimates for the participant's actions under both models. This allows us to calculate the confusability score of the participant's action as $p(A_i = \mathrm{A}|A_{-i}, B)$ with A_i being the particular action chosen by a participant playing level A, A_{-i} being all actions except the participant's action, and B being all actions from participants in level B. We obtain the same confusability metric for participants in level B by swapping A and B. Finally, we average the confusability across both variants to calculate how differentiable actions were on a template.

If these scores are greater than 0.5, it implies that the actions are different across the matched levels because we can determine which level caused which action. We find that participants' actions are differentiable in all levels except for 'Towers' and 'Falling' (see Table A.1). In 'Falling', this is because people have a strong prior to drop objects from above, so blocking placements below the cup has little effect (see the main paper results for further discussion). In 'Towers', people do not seem to initially understand that they must hit the red block from the left in 'Towers B' and so in both settings they just try to disturb the pile of blocks.

We can perform a similar analysis for the SSUP and DQN models (Table A.1), treating each model run as a different participant. We find that the SSUP model does take different actions on all of the levels, but the DQN model does not. We compare to the DQN *without* updating, to demonstrate that the learned policy does not notice these subtle differences in the scenes.

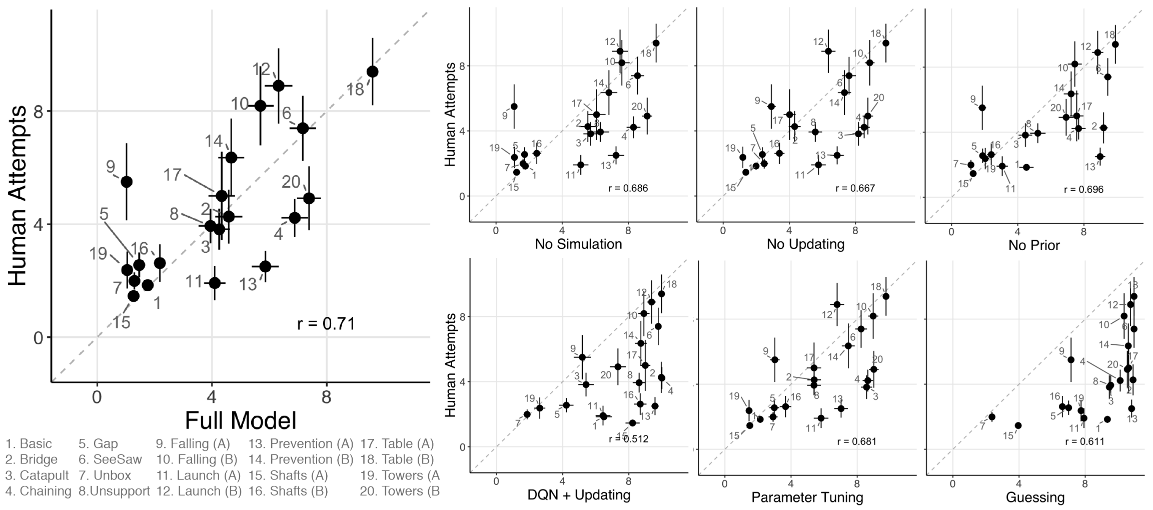


Figure A-4: Comparison of average number of human participants' attempts for each level with average number of attempts for the full model (*left*) and six alternate models. Bars indicate 95% confidence intervals on estimates of the means. The number of placements was capped at 10 for all models. If a model took more than 10 attempts on a particular level, it is considered unsolved. Model results are combined over 250 runs.

A.6.2 Comparisons across models

Figure A-4 shows the correlations between alternate models and human performance for the average number of placements across levels from the main experiment. Figure A-5 shows correlations between alternate models and humans for both accuracy and placements in the validation experiment.

Table A.2 shows the RMSE comparing each of the models with human data from the main experiment, and Table A.3 shows the same information for the validation experiment.

Table A.4 shows the ABCS scores for each level individually. By examining which levels the varying alternate models perform well in, it is clear what the relative contributions

Table A.1: Confusability metric between matched level pairs. Brackets indicate bootstrapped 95% confidence intervals on the estimate.

Trial	Humans	SSUP	DQN+Updating
Shafts	0.72 [0.66, 0.78]	0.77 [0.73, 0.83]	0.50 [0.47, 0.52]
Prevention	0.54 [0.51, 0.58]	0.53 [0.50, 0.57]	0.51 [0.49, 0.54]
Launch	0.61 [0.56, 0.68]	0.60 [0.56, 0.63]	0.51 [0.49, 0.54]
Falling	0.48 [0.42, 0.54]	0.92 [0.89, 0.95]	0.51 [0.49, 0.54]
Table	0.56 [0.51, 0.61]	0.74 [0.69, 0.79]	0.49 [0.47, 0.52]
Towers	0.47 [0.41, 0.52]	0.64 [0.60, 0.68]	0.48 [0.46, 0.51]

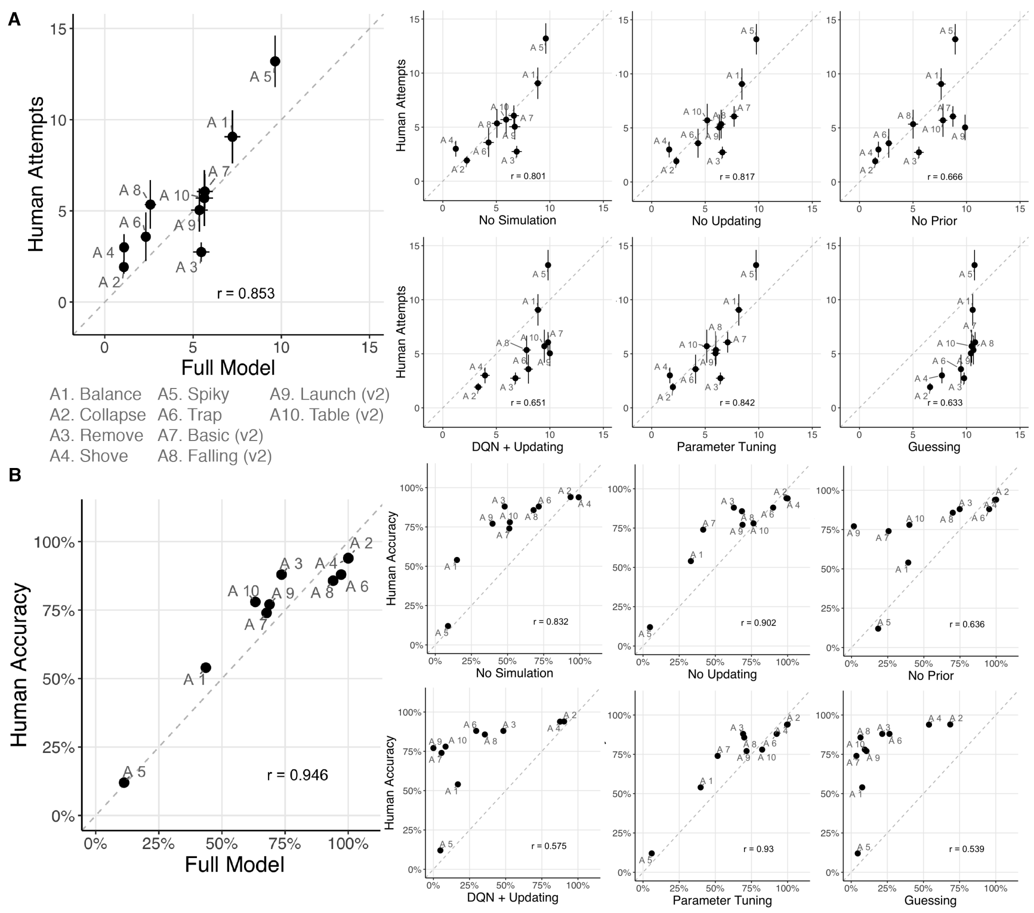


Figure A-5: Comparisons of human vs. all model performance for the validation experiment. **(A)** Human vs. model placements per trial. **(B)** Human vs. model trial accuracy. Bars indicate 95% confidence intervals on estimates of the means.

Table A.2: Comparisons between people and all models on the original experiment. Brackets indicate bootstrapped 95% confidence intervals on the estimate. 'Average ABCS' refers to the average of the *Area Between Cumulative Solution* curves of participants and the model. 'Guessing' model placements and accuracy can be calculated exactly and therefore have no confidence intervals.

Model	Avg. Attempts	Attempt RMSE	Accuracy	Accuracy RMSE	Average ABCS
Human	4.48 [4.25, 4.66]	-	0.81	-	-
Full Model	4.24 [4.17, 4.32]	1.86 [1.66, 2.17]	0.77 [0.76, 0.78]	0.135 [0.121, 0.169]	0.111 [0.103, 0.132]
No Updating	5.38 [5.32, 5.46]	2.31 [2.16, 2.59]	0.69 [0.68, 0.7]	0.271 [0.248, 0.299]	0.189 [0.173, 0.208]
No Simulation	5.23 [5.14, 5.31]	2.29 [2.12, 2.58]	0.59 [0.58, 0.6]	0.328 [0.307, 0.353]	0.218 [0.204, 0.237]
No Prior	5.55 [5.48, 5.62]	2.45 [2.29, 2.71]	0.61 [0.6, 0.62]	0.305 [0.288, 0.33]	0.202 [0.187, 0.222]
DQN + Updating	7.52 [7.44, 7.61]	3.89 [3.76, 4.1]	0.34 [0.33, 0.35]	0.544 [0.527, 0.566]	0.397 [0.381, 0.416]
Parameter Tuning	5.7 [5.64, 5.78]	2.39 [2.25, 2.65]	0.65 [0.64, 0.66]	0.293 [0.275, 0.323]	0.194 [0.180, 0.214]
Guessing	8.88	4.91 [4.75, 5.1]	0.32	0.552 [0.531, 0.573]	0.402 [0.385, 0.417]

of the prior, simulator and update mechanism are. For example, in Catapult, where updating an action to move along a gradient is useful, the "No Updating" ablation performs worst of all methods. Similarly for Bridge, where the prior focuses the search particularly well but a structured search is less necessary (since almost all actions with a positive outcome lead to solutions), removing the object-oriented prior is most problematic for fitting human data. These results provide intuition for why the different model components are important for explaining rapid trial-and-error learning.

Although all alternate models have higher ABCS scores than the full SSUP model, we can test whether these are statistically reliable differences. We form a null hypothesis distribution by bootstrapping the difference between the ABCS scores of two runs of the full SSUP model, taking $10,000$ samples. We then test where the differences between the scores of the full and alternate models fall in that distribution. On the main levels, we find that the differences between ABCS scores for all models would be extremely unlikely (the values were more extreme than *all* null hypothesis samples; all $ps < 0.0001$). On the validation levels, we find that the models that remove the prior or simulation engine, or replace the prior with a DQN all have ABCS score far outside the expectations from the null hypothesis distribution (all $ps < 0.0001$), but the ABCS scores of the models without an update mechanism or with parameter tuning in place of the update mechanism fell within the null hypothesis distribution (no updating: $p = 0.055$, parameter tuning: $p = 0.346$).

Table A.3: Comparisons between people and all models on the validation experiment. Brackets indicate bootstrapped 95% confidence intervals on the estimate. 'Average ABCS' refers to the average of the *Area Between Cumulative Solution* curves of participants and the model. 'Guessing' model placements and accuracy can be calculated exactly and therefore have no confidence intervals.

Model	Avg. Attempts	Attempt RMSE	Accuracy	Accuracy RMSE	Average ABCS
Human	5.57 [5.11, 6.03]	-	0.74	-	-
Full Model	4.61 [4.5, 4.73]	1.93 [1.64, 2.4]	0.72 [0.71, 0.73]	0.0933 [0.0708, 0.145]	0.0936 [0.080, 0.124]
No Updating	5.88 [5.77, 5.98]	1.88 [1.65, 2.33]	0.65 [0.63, 0.66]	0.162 [0.135, 0.209]	0.120 [0.105, 0.152]
No Simulation	5.75 [5.62, 5.88]	1.93 [1.71, 2.36]	0.55 [0.53, 0.56]	0.252 [0.215, 0.295]	0.170 [0.150, 0.200]
No Prior	5.95 [5.84, 6.07]	2.56 [2.25, 3.01]	0.56 [0.55, 0.58]	0.321 [0.284, 0.364]	0.201 [0.178, 0.228]
DQN + Updating	7.79 [7.69, 7.89]	3.31 [2.93, 3.75]	0.33 [0.32, 0.34]	0.494 [0.456, 0.534]	0.344 [0.320, 0.374]
Parameter Tuning	5.63 [5.52, 5.73]	1.76 [1.52, 2.18]	0.68 [0.67, 0.7]	0.121 [0.0953, 0.17]	0.1 [0.086, 0.130]
Guessing	9.72	4.89 [4.62, 5.22]	0.21	0.575 [0.544, 0.61]	0.422 [0.397, 0.451]

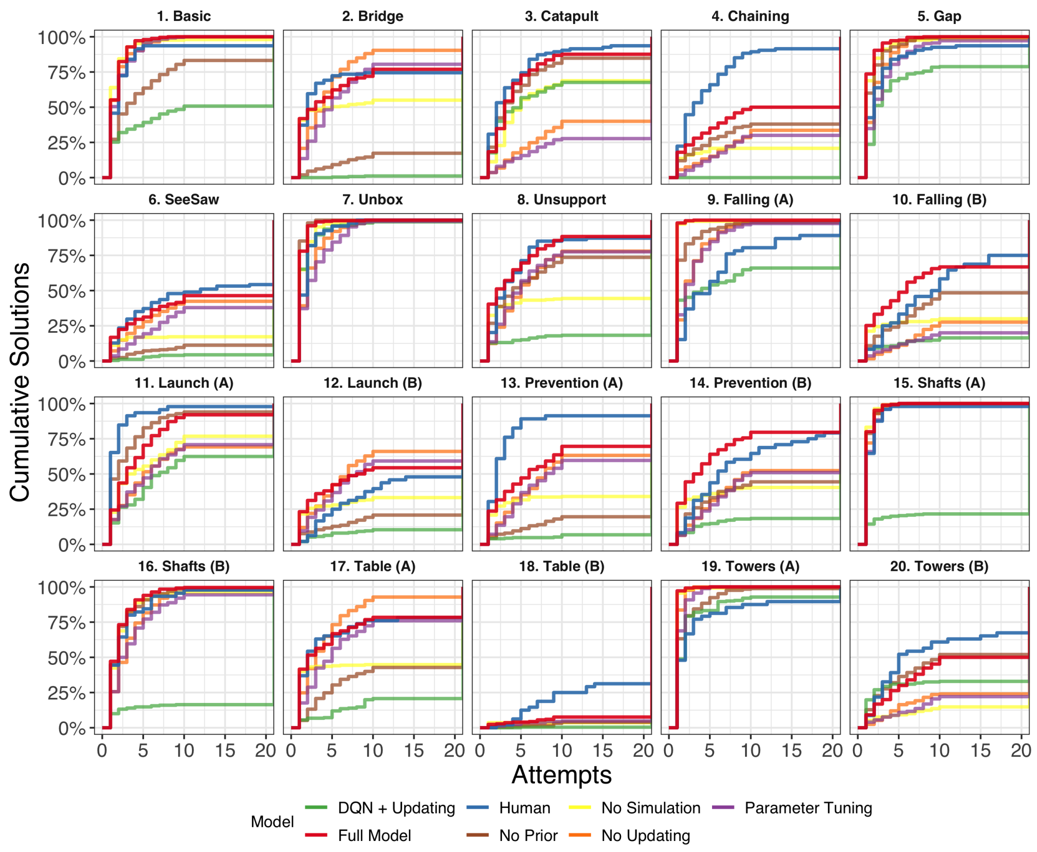


Figure A-6: Cumulative solution rates for all alternate models on the main experiment levels.

Since the validation levels were not explicitly chosen to differentiate models, only one level, "Basic (v2)", would be expected to benefit from a policy updating mechanism, as hitting the ball from higher is necessary to succeed. Indeed, this is the one level where we do see an improvement from the SSUP model as compared to to the Parameter Tuning alternative.

A.6.3 Likelihoods of placements

In addition to testing whether each of the models capture the evolution of human successes and failure over the course of each trial, we can study how well each model captures the specific actions that people take. We measure this as the likelihood of an action under the model's predictions.

However, this measure not perfectly reliable past the first action for two reasons. First,

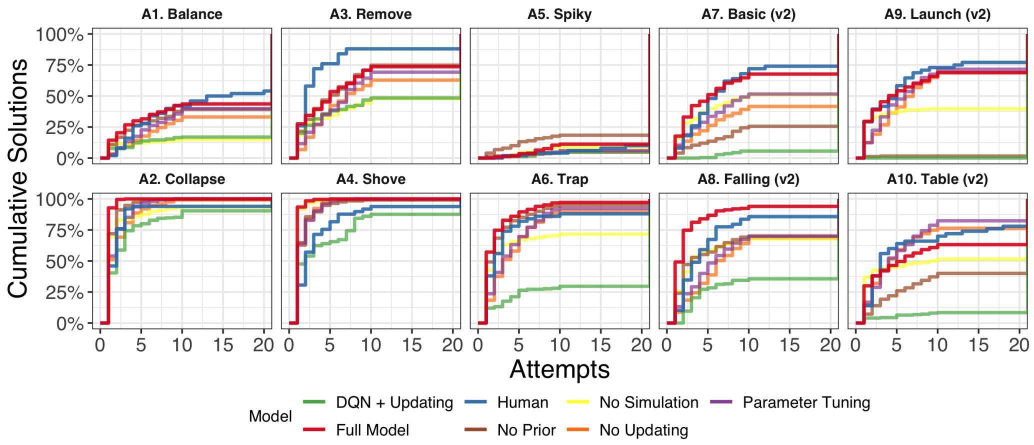


Figure A-7: Cumulative solution rates for all alternate models on the validation levels.

the model and people take actions that depend on the results of prior actions, and therefore the probability density for the second action calculated by the model is conditional on the *model's* prior actions, not *people's*. Second, because of varying solution rates, the reliability of the probability density and the likelihood will vary by trial – for instance, only 2% of model runs on 'Falling (A)' even take a second action, and only a third of participants did not solve 'Shafts (A)' on the first attempt, and these numbers grow smaller with subsequent action orders. We therefore measure this likelihood in two cases, to mirror Fig. A-11: the likelihood of the first attempt (where these problems do not exist), and the likelihood of the final solution, to estimate how well the models capture the pattern of final solutions.

Because we only obtain samples from the model, we cannot calculate the likelihood function directly but instead must approximate it via a kernel density estimation (KDE) based around each of the sampled points. This requires two additional parameters to fit: (1) the width of the kernel, and (2) a background "guessing" rate. We fit these parameters for each model to maximize likelihood on the first placement of the basic levels, but use those same fit parameters to calculate the likelihood of placements on the validation levels, and of the last placements on the basic levels.

In addition, we calculate a noise ceiling by asking how well participants' placements can be explained by all other participants' actions on that level. We use the same KDE procedure described above, fit using leave-one-out validation and treating other participants'

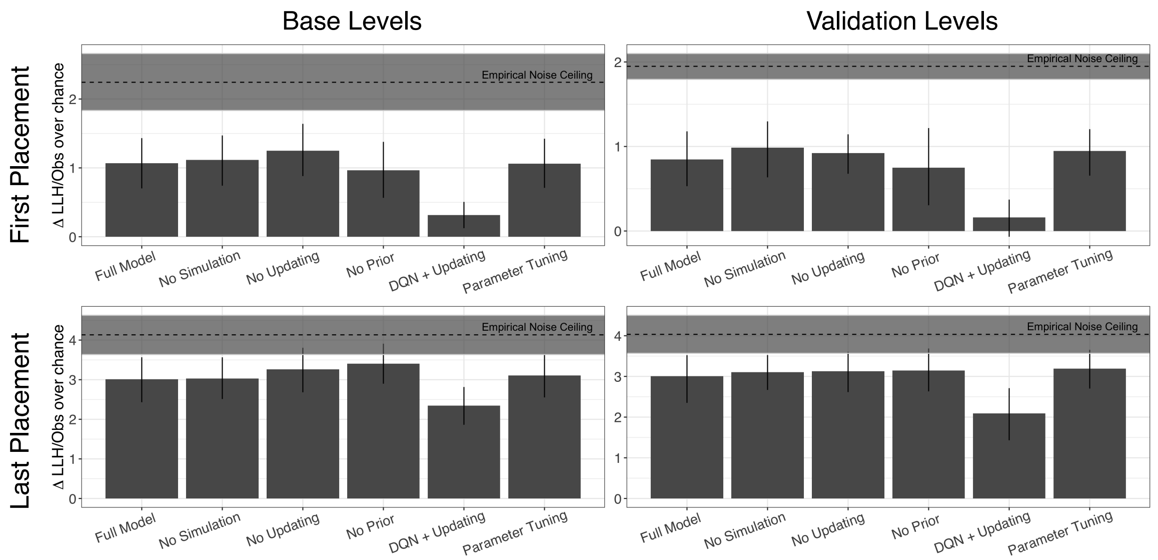


Figure A-8: Improvement in log-likelihoods per observation over chance placements for the first placement of a trial (*top*) and final solution placement (*bottom*), for the 20 regular levels (*left*) and 10 validations levels (*right*). Bars represent 95% bootstrapped CIs. The dashed line represents the maximum noise ceiling calculated as likelihood of placements given all other participants' placements, with the grey area representing the 95% CI on that metric.

actions similar to model observations. We also calculated the likelihood under a model that took random actions, so that we could scale these model fits between chance and the noise ceiling.

Finally, to test how stable these fits are, we calculated bootstrapped 95% confidence intervals on the likelihoods. We bootstrap by trial to control for particular levels where one model might happen to excel or fall short due to chance sampling of actions, then weight the log-likelihoods by the number of human observations to avoid unbalancing the likelihoods by this scheme.

The results of this analysis can be seen in Fig. A-8. We find that this is a noisy measure (due largely in unexplanable variability in participants' actions; see Figs. A-11, A-12), and so cannot reliably differentiate any of the models' performance, with the exception of the 'DQN + Updating' model which does reliably worse than all the rest. However, we do find that *all* models fall short of the noise ceiling, suggesting that there are additional features of human trial-and-error problem solving that these models are not capturing (see the Discussion for some proposals).

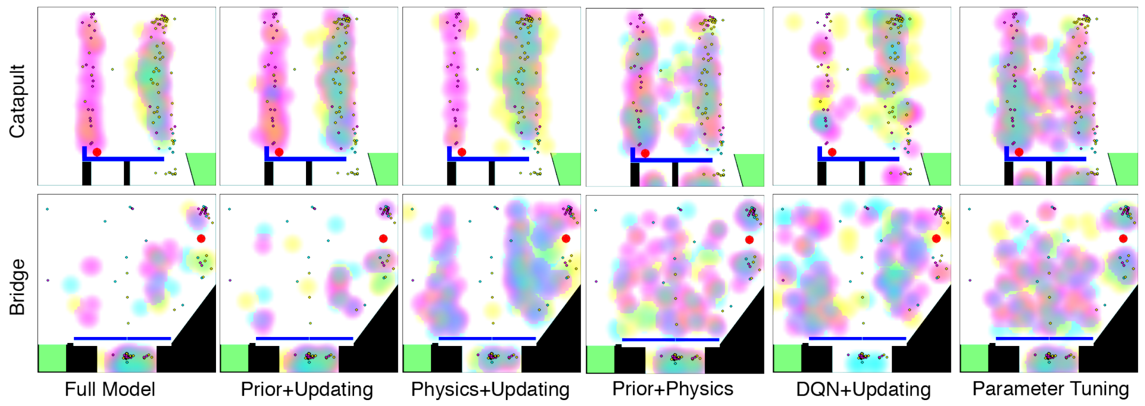


Figure A-9: Comparisons of model predictions across "Catapult" and "Bridge" for the first action. In "Catapult", the full model has the best likelihood, as it is able to narrow down the set of actions into those that are most promising - like people. In "Bridge", the "Prior+Physics" and "Parameter Tuning" ablations perform best in terms of likelihood as they have higher coverage for human participants that took actions above the bridge.

We can see this in more detail by looking at the first actions taken by each alternative model in two representative trials: "Bridge" and "Catapult" (Fig. A-9). In "Catapult", the SSUP model and alternatives which include shaping around *reward* correctly narrow down the set of possible actions towards the same set that people consider. Ablations without this reward narrowing perform worse in terms of likelihood of human actions.

In "Bridge", the SSUP model and ablations which are guided by the reward narrow the action space *too* much with respect to the heterogeneity of human actions. The likelihood of "Prior+Physics" and "Parameter tuning" is therefore better in these cases.

A.6.4 Analyzing switching behavior

We looked at whether the model was capturing the likelihood of participants "switching" from one strategy to another. In general, it is not obvious what constitutes a particular strategy for participants. Strategies are intentional - just because you did not successfully hit the ball towards the right does not mean that you did not plan to hit the ball such that it moved towards the right. To approximately quantify strategy switching, we therefore consider a "switch" to have occurred if the tool changed, or if the closest object to the performed action changed. Although this does not capture the the full richness of human "strategies," here we use these measures as a useful proxy and leave precise definitions of this notion to future work.

We quantified how often participants and the model switched strategies under this definition. For both tool switches and relative object switches, we find a good correlation between the human participants and the model (see Figure A-10).

Finally, we provide additional results showing the first and last placements of the model relative to participants for all levels from the main experiment (Figure A-11, as well as for the validation levels, Figure A-12). In most cases, these show qualitatively similar initial attempts, and evolutions towards a solution (however, see the Results in the main text for discussion of exceptions).

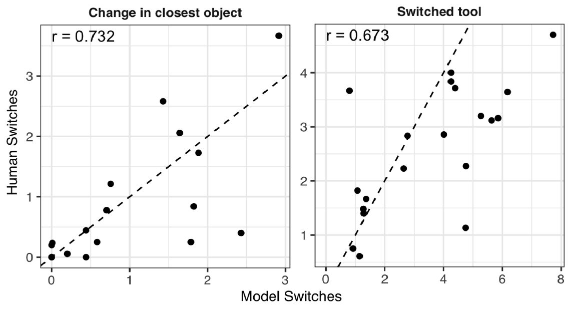


Figure A-10: Comparison of human participants' average number of strategy switches on each trial versus the average number of strategy switches of the full model.

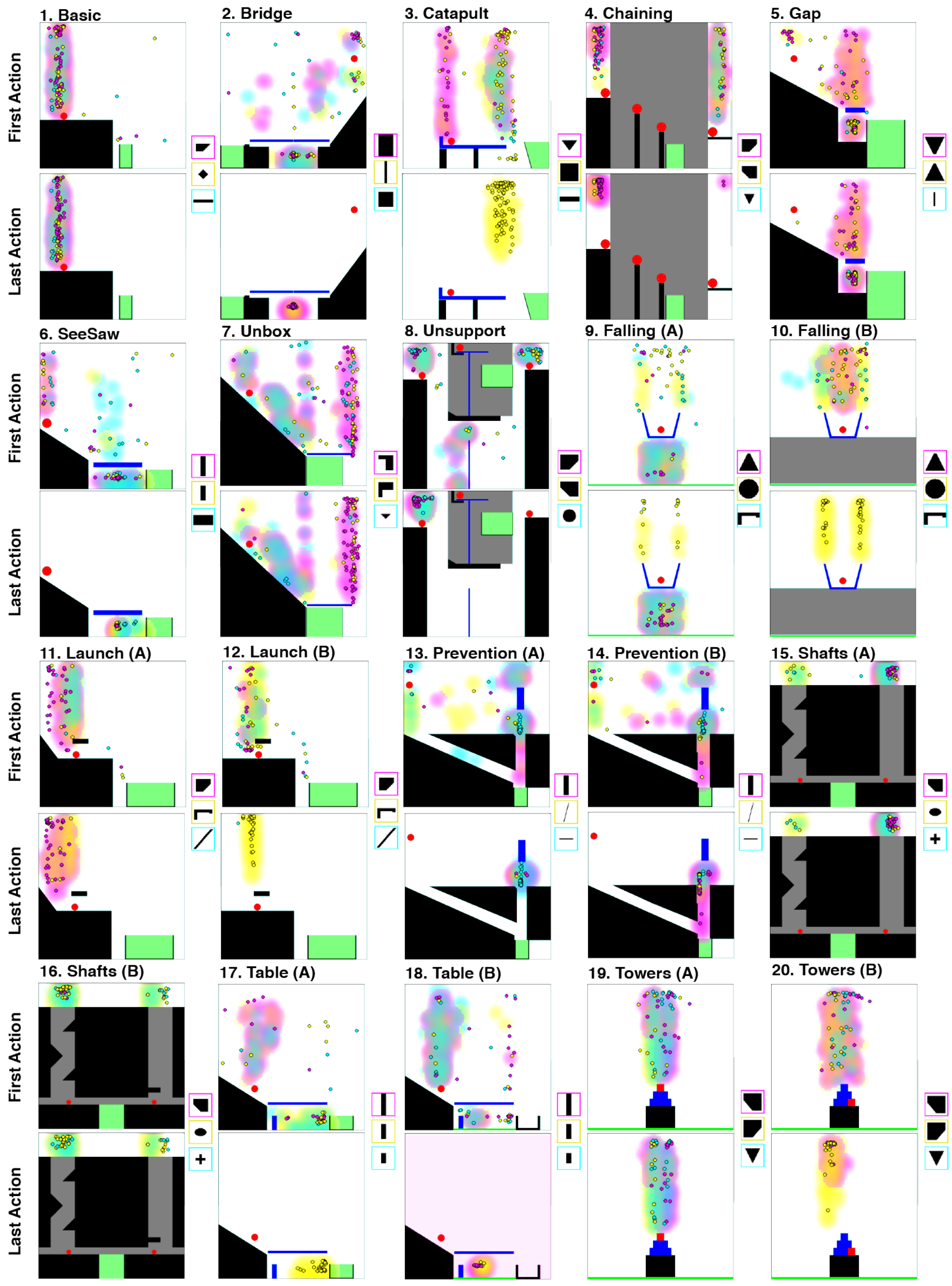


Figure A-11: Distribution of predicted model actions (background) versus human actions (points) on the first attempts of the level (*top*) and the attempt used to solve the level (*bottom*) across all levels from Experiment 1.

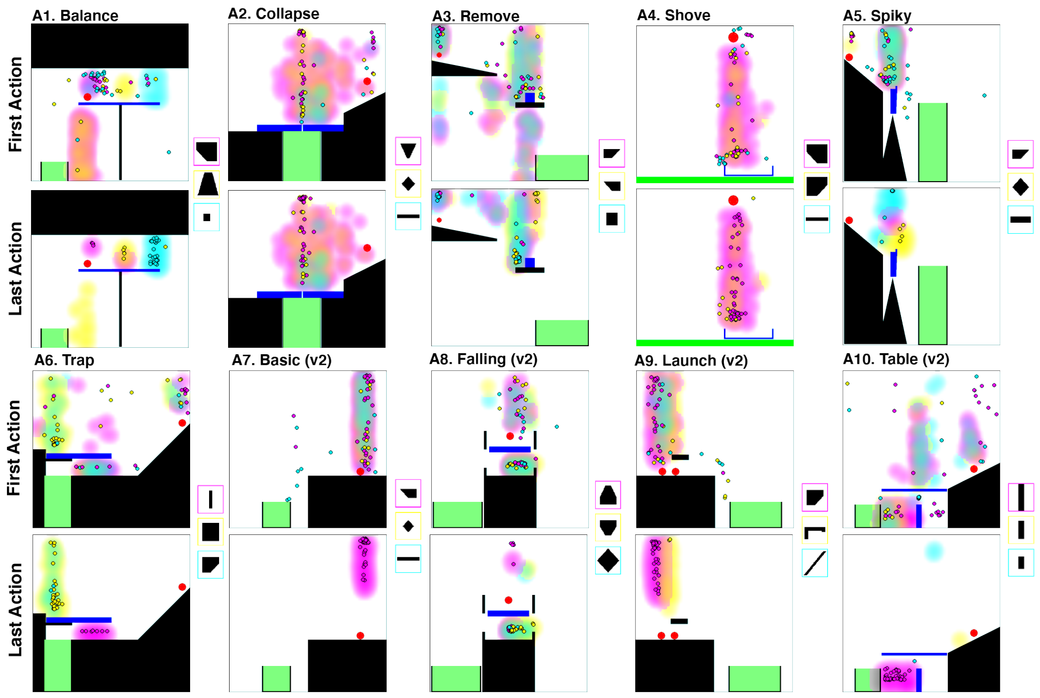


Figure A-12: Distribution of predicted model actions (background) versus human actions (points) on the first attempts of the level (*top*) and the attempt used to solve the level (*bottom*) across all levels from Experiment 2.

	SSUP	No Updating	No Simulator	No Prior	DQN + Upd	Param Tune
Basic	0.063	0.053	0.052	0.157	0.420	0.048
Bridge	0.045	0.134	0.179	0.546	0.668	0.101
Catapult	0.057	0.493	0.225	0.081	0.217	0.576
Chaining	0.340	0.504	0.561	0.441	0.750	0.528
Gap	0.099	0.053	0.079	0.080	0.143	0.038
SeeSaw	0.052	0.090	0.267	0.341	0.393	0.143
Unbox	0.033	0.025	0.020	0.039	0.019	0.048
Unsupport	0.030	0.117	0.340	0.130	0.576	0.111
Falling (A)	0.261	0.162	0.253	0.213	0.155	0.153
Falling (B)	0.097	0.304	0.258	0.137	0.365	0.349
Launch (A)	0.137	0.335	0.265	0.077	0.403	0.329
Launch (B)	0.121	0.180	0.102	0.182	0.255	0.124
Prevention (A)	0.250	0.332	0.503	0.660	0.756	0.358
Prevention (B)	0.114	0.155	0.226	0.195	0.400	0.169
Shafts (A)	0.029	0.024	0.030	0.027	0.713	0.018
Shafts (B)	0.034	0.035	0.022	0.017	0.712	0.064
Table (A)	0.016	0.123	0.261	0.340	0.524	0.062
Table (B)	0.147	0.173	0.173	0.176	0.198	0.165
Towers (A)	0.154	0.147	0.148	0.100	0.041	0.134
Towers (B)	0.135	0.335	0.402	0.107	0.237	0.358
Average	0.111	0.189	0.218	0.202	0.397	0.194
Balance	0.055	0.124	0.250	0.075	0.234	0.074
Collapse	0.086	0.047	0.032	0.067	0.071	0.050
Remove	0.173	0.282	0.374	0.170	0.367	0.240
Shove	0.136	0.110	0.126	0.099	0.081	0.105
Spiky	0.027	0.018	0.018	0.097	0.018	0.012
Trap	0.095	0.062	0.137	0.065	0.530	0.068
Basic (v2)	0.061	0.242	0.164	0.374	0.531	0.163
Falling (v2)	0.140	0.180	0.139	0.130	0.416	0.151
Launch (v2)	0.059	0.098	0.270	0.619	0.634	0.067
Table (v2)	0.104	0.038	0.189	0.313	0.559	0.072
Average	0.094	0.120	0.170	0.201	0.344	0.100

Table A.4: Area Between Cumulative Solutions metrics for each level (row) by model (column). *Top:* 20 levels from the main experiments. *Bottom:* 10 levels from the validation experiment.

Bibliography

[1] Brain it on. https://brainitongame.com/, 2015.

[2] Salvatore M Aglioti, Paola Cesari, Michela Romani, and Cosimo Urgesi. Action anticipation and motor resonance in elite basketball players. *Nature neuroscience*, 11(9):1109, 2008.

[3] Kelsey Allen, Kevin Smith, and Josh Tenenbaum. Rapid trial-and-error learning in physical problem solving. *Structure and Priors in Reinforcement Learning Workshop, International Conference on Learning Representations*, 2019.

[4] Kelsey R Allen, Evan Shelhamer, Hanul Shin, and Joshua B Tenenbaum. Infinite mixture prototypes for few-shot learning. 2019.

[5] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. 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, 2020.

[6] John R Anderson. Problem Solving and Learning. *Am Psychol*, page 10, 1993.

[7] John R Anderson. Problem solving. In *Cognitive Psychology and Its Implications*, chapter 8, pages 181–209. Worth Publishers, New York, NY, 8th edition, 2015.

[8] Arnaud Badets and François Osiurak. A goal-based mechanism for delayed motor intention: Considerations from motor skills, tool use and action memory. *Psychological Research*, 79(3):345–360, 2015.

[9] Renée Baillargeon, Jie Li, Yael Gertner, and Di Wu. How do infants reason about physical events? In *The Wiley-Blackwell Handbook of Childhood Cognitive Development*, pages 11–48. Wiley-Blackwell, 2011.

[10] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *International Conference on Learning Representations*, 2019.

[11] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. volume 32, pages 5082–5093, 2019.

[12] Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica Hamrick. Structured agents for physical construction. In *International Conference on Machine Learning*, pages 464–474, 2019.

[13] Christopher Bates, Peter W Battaglia, Ilker Yildirim, and Joshua B Tenenbaum. Humans predict liquid dynamics using probabilistic simulation. In *Cognitive Science Society*, pages 172–177, 2015.

[14] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

[15] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *PNAS*, 110(45):18327–18332, 2013.

[16] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4509–4517, 2016.

[17] Sarah R Beck, Ian A Apperly, Jackie Chappell, Carlie Guthrie, and Nicola Cutting. Making tools isn't child's play. *Cognition*, 119(2):301–306, 2011.

[18] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

[19] Apratim Bhattacharyya, Mateusz Malinowski, Bernt Schiele, and Mario Fritz. Long-term image boundary prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] Anna M Borghi. Object concepts and action: Extracting affordances from objects parts. *Acta Psychologica*, 115(1):69–96, 2004.

[21] Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[22] Mark E Bouton. Context and behavioral processes in extinction. *Learning & memory*, 11(5):485–494, 2004.

[23] Robert Boyd and Peter J Richerson. Why culture is common, but cultural evolution is rare. In *Proceedings-British Academy*, volume 88, pages 77–94. OXFORD UNIVERSITY PRESS INC., 1996.

[24] Ann L Brown. Domain-specific principles affect learning and transfer in children. *Cognitive science*, 14(1):107–133, 1990.

[25] Laurel J Buxbaum. Ideomotor apraxia: a call to action. *Neurocase*, 7(6):445–458, 2001.

[26] Laurel J Buxbaum and Solène Kalénine. Action knowledge, visuomotor activation, and embodiment in the two action systems. *Annals of the New York Academy of Sciences*, 1191:201, 2010.

[27] Kayleigh Carr, Rachel L Kendal, and Emma G Flynn. Eureka!: What is innovation, how does it develop, and who does it? *Child development*, 87(5):1505–1519, 2016.

[28] Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A Compositional Object-Based Approach to Learning Physical Dynamics. 2016.

[29] Linda L Chao and Alex Martin. Representation of manipulable man-made objects in the dorsal stream. *Neuroimage*, 12(4):478–484, 2000.

[30] HeeSun Choi, Cindy Crump, Christian Duriez, Asher Elmquist, Gregory Hager, David Han, Frank Hearl, Jessica Hodgins, Abhinandan Jain, Frederick Leve, et al. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1), 2021.

[31] Yun Chu and James N MacGregor. Human performance on insight problem solving: A review. *The Journal of Problem Solving*, 3(2):6, 2011.

[32] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[33] Allan M Collins and Elizabeth F Loftus. A spreading-activation theory of semantic processing. *Psychological review*, 82(6):407, 1975.

[34] Anne Collins and Etienne Koechlin. Reasoning, learning, and creativity: frontal lobe function and human decision-making. *PLoS Biol*, 10(3):e1001293, 2012.

[35] Anne Gabrielle Eva Collins. Reinforcement learning: bringing together computation and cognition. *Current Opinion in Behavioral Sciences*, 29:63–68, 2019.

[36] Anne GE Collins and Michael J Frank. Cognitive control over learning: Creating, clustering, and generalizing task-set structure. *Psychological review*, 120(1):190, 2013.

[37] K. J. W. Craik. *The Nature of Explanation*. CUP Archive, 1943. Google-Books-ID: wT04AAAAIAAJ.

[38] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33, 2020.

[39] Marco F. Cusumano-Towner, Feras A. Saad, Alexander Lew, and Vikash K. Mansinghka. Gen: A general-purpose probabilistic programming system with programmable inference. Technical Report MIT-CSAIL-TR-2018-020, Cambridge, Massachusetts, November 2018.

[40] Nicola Cutting, Ian A Apperly, and Sarah R Beck. Why do children lack the flexibility to innovate tools? *Journal of experimental child psychology*, 109(4):497–511, 2011.

[41] Nicola Cutting, Ian A. Apperly, Jackie Chappell, and Sarah R. Beck. The puzzling difficulty of tool innovation: Why can't children piece their knowledge together? *Journal of Experimental Child Psychology*, 125:110–117, 2014.

[42] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems*, volume 12, page 00052. Ann Arbor, MI, USA, 2016.

[43] Ishita Dasgupta, Kevin A. Smith, Eric Schulz, Joshua B. Tenenbaum, and Samuel J. Gershman. Learning to act by integrating mental simulations and physical experiments. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, 2018.

[44] Ernest Davis. Pouring liquids: A study in commonsense physical reasoning. *Artificial Intelligence*, 172(12-13):1540–1578, 2008.

[45] Ernest Davis. How does a box work? A study in the qualitative dynamics of solid objects. *Artificial Intelligence*, 175(1):299–345, 2010.

[46] Ernest Davis, Gary Marcus, and Angelica Chen. Reasoning from Radically Incomplete Information: The Case of Containers. *Advances in Cognitive Systems*, 2:1–18, 2013.

[47] Johan De Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial intelligence*, 24(1-3):7–83, 1984.

[48] Lewis G Dean, Gill L Vale, Kevin N Laland, Emma Flynn, and Rachel L Kendal. Human cumulative culture: a comparative perspective. *Biological Reviews*, 89(2):284–301, 2014.

[49] Margaret Anne Defeyter and Tim P German. Acquiring an understanding of design: evidence from children's insight problem solving. *Cognition*, 89(2):133–155, 2003.

[50] Marc Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–142, 2013.

[51] Misha Denil, Pulkit Agrawal, Tejas D. Kulkarni, Tom Erez, Peter W. Battaglia, and Nando de Freitas. Learning to Perform Physics Experiments via Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2017.

[52] Sašo Džeroski, Luc De Raedt, and Hendrik Blockeel. Relational reinforcement learning. In *International Conference on Inductive Logic Programming*, pages 11–22. Springer, 1998.

[53] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

[54] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[55] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[56] Jason Fischer, John G. Mikhael, Joshua B. Tenenbaum, and Nancy Kanwisher. Functional neuroanatomy of intuitive physical inference. *Proceedings of the National Academy of Sciences*, 113(34):E5072–E5081, 2016.

[57] Kenneth D Forbus. Spatial and qualitative aspects of reasoning about motion. In *AAAI*, volume 80, pages 170–173, 1980.

[58] Kenneth D Forbus. Qualitative Reasoning About Space and Motion. In *Mental Models*, chapter 4, pages 53–73. 1983.

[59] Kenneth D Forbus. Qualitative physics: Past, present, and future. In *Exploring artificial intelligence*, pages 239–296. Elsevier, 1988.

[60] Kenneth D Forbus. Qualitative modeling. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(4):374–391, jul 2011.

[61] Kenneth D Forbus and Dedre Gentner. Learning physical domains: Towards a theoretical framework. In *Machine Learning: An Artificial Intelligence Approach*. Tioga press, 1986.

[62] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. In *International Conference on Learning Representations*, 2016.

[63] Michael C Frank and David Barner. Representing exact number visually using mental abacus. *Journal of Experimental Psychology: General*, 141:134–149, 2011.

[64] Nicholas T Franklin and Michael J Frank. Compositional clustering in task structure learning. *PLoS computational biology*, 14(4):e1006116, 2018.

[65] Peter A Frensch and Joachim Funke. *Complex problem solving: The European perspective*. Psychology Press, 1995.

[66] Scott Friedman and Kenneth D Forbus. Learning naïve physics models and misconceptions. In *Proceedings of the annual conference of the cognitive science society*, pages 2505–2510, 2009.

[67] Jason P Gallivan, Craig S Chapman, Daniel M Wolpert, and J Randall Flanagan. Decision-making in sensorimotor control. *Nature Reviews Neuroscience*, 19(9):519–534, 2018.

[68] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[69] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2021.

[70] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 37(13-14):1796–1825, 2018.

[71] Xiaoyu Ge, Jae Hee Lee, Jochen Renz, and Peng Zhang. Hole in one: Using qualitative reasoning for solving hard physical puzzle problems. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1762–1763. IOS Press, 2016.

[72] Joseph Gentle. ChipmunkJS, 2017.

[73] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

[74] Samuel J. Gershman, Arthur B. Markman, and A. Ross Otto. Retrospective revaluation in sequential decision making: A tale of two systems. *J Exp Psychol Gen*, 143(1):182–194, 2014.

[75] Samuel J. Gershman, Jimmy Zhou, and Cody Kommers. Imaginative Reinforcement Learning: Computational Principles and Neural Mechanisms. *J Cognitive Neurosci*, 29(12):2103–2113, 2017.

[76] S.J. Gershman, K.A. Norman, and Y. Niv. Discovering latent causes in reinforcement learning. *Curr. Opin. Behav. Sci*, 5:43–50, 2015.

[77] Tobias Gerstenberg, Noah D Goodman, David A Lagnado, and Joshua B Tenenbaum. Noisy Newtons: Unifying process and dependency accounts of causal attribution. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2012.

[78] Tobias Gerstenberg, Noah D. Goodman, David A. Lagnado, and Joshua B Tenenbaum. How, whether, why: Causal judgments as counterfactual contrasts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, page 6, 2015.

[79] Tobias Gerstenberg, Matthew F. Peterson, Noah D. Goodman, David A. Lagnado, and Joshua B. Tenenbaum. Eye-Tracking Causality. *Psychological Science*, 28(12):1731–1744, 2017.

[80] Tobias Gerstenberg, Max H. Siegel, and Josh B. Tenenbaum. What happened? reconstructing the past through vision and sound. proceedings of the 40th annual conference of the cognitive science society. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2018.

[81] James J Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin Co., 1979.

[82] Mary L. Gick and Keith J. Holyoak. Analogical problem solving. *Cognitive Psychol*, 12(3):306–355, 1980.

[83] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

[84] G. Goldenberg and J. Spatt. The neural basis of tool use. *Brain*, 132(6):1645–1655, 2009.

[85] Georg Goldenberg and Sonja Hagmann. Tool use and mechanical problem solving in apraxia. *Neuropsychologia*, 36(7):581–589, 1998.

[86] Leslie J Gonzalez Rothi, Cynthia Ochipa, and Kenneth M Heilman. A cognitive neuropsychological model of limb praxis. *Cognitive Neuropsychology*, 8(6):443–458, 1991.

[87] Alison Gopnik. Childhood as a solution to explore–exploit tensions. *Philosophical Transactions of the Royal Society B*, 375(1803):20190502, 2020.

[88] Alison Gopnik, David M Sobel, Laura E Schulz, and Clark Glymour. Causal learning mechanisms in very young children: two-, three-, and four-year-olds infer causal relations from patterns of variation and covariation. *Developmental psychology*, 37(5):620, 2001.

[89] Patricia M Greenfield. Language, tools and brain: The ontogeny and phylogeny of hierarchically organized sequential behavior. *Behavioral and brain sciences*, 14(4):531–551, 1991.

[90] Thomas L Griffiths, Kevin R Canini, Adam N Sanborn, and Daniel J Navarro. Unifying rational models of categorization via the hierarchical dirichlet process. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2007.

[91] Thomas L. Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua B. Tenenbaum. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14:357–364, 2010.

[92] Thomas L. Griffiths, Mark Steyvers, and Alana Firl. Google and the mind: Predicting fluency with pagerank. *Psychological Science*, 18(12):1069–1076, 2007.

[93] Todd M. Gureckis, Jay Martin, John McDonnell, Alexander S. Rich, Doug Markant, Anna Coenen, David Halpern, Jessica B. Hamrick, and Patricia Chan. psiTurk: An open-source framework for conducting replicable behavioral experiments online. *Behavior Research Methods*, 48(3):829–842, 2016.

[94] Nobuhiro Hagura, Patrick Haggard, and Jörn Diedrichsen. Perceptual decisions are biased by the cost to act. *Elife*, 6:e18422, 2017.

[95] Ibrahim Abou Halloun and David Hestenes. Common sense concepts about motion. *American Journal of Physics*, 53(11):1056–1065, 1985.

[96] Jessica B Hamrick, Andrew J Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W Battaglia. Metacontrol for adaptive imagination-based optimization. *International Conference on Learning Representations*, 2017.

[97] Jessica B. Hamrick, Peter W. Battaglia, Thomas L. Griffiths, and Joshua B. Tenenbaum. Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76, 2016.

[98] Jessica B. Hamrick, Kevin A. Smith, Thomas L. Griffiths, and Edward Vul. Think again? The amount of mental simulation tracks uncertainty in the outcome. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, 2015.

[99] Christopher M Harris and Daniel M Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784, 1998.

[100] Patrick J. Hayes. The naive physics manifesto. In Donald Michie, editor, *Expert Systems in the Micro-electronic Age*, pages 242–270. Edinburgh University Press, 1979.

[101] James B Heald, James N Ingram, J Randall Flanagan, and Daniel M Wolpert. Multiple motor memories are learned to control different points on a tool. *Nature human behaviour*, 2(4):300–311, 2018.

[102] Heiko Hecht and Marco Bertamini. Understanding projectile acceleration. *Journal of Experimental Psychology: Human Perception and Performance*, 26(2):730–746, 2000.

[103] Mary Hegarty. Mental Animation: Inferring Motion From Static Displays of Mechanical Systems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(5):1084–1102, 1992.

[104] Mary Hegarty. Mechanical reasoning by mental simulation. *Trends in Cognitive Sciences*, 8(6):280–285, 2004.

[105] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[106] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012.

[107] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[108] Keith Holyoak. Analogy and relational reasoning. In Keith J. Holyoak and R. G. Morrison, editors, *The Oxford Handbook of Thinking and Reasoning*, pages 234–259. 2012.

[109] B. Hommel. Inverting the simon effect by intention. *Psychological Research*, 55(270 –279), 1993.

[110] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Computer Vision and Pattern Recognition*, 2019.

[111] Helen J Huang, Rodger Kram, and Alaa A Ahmed. Reduction of metabolic cost during motor learning of arm reaching dynamics. *Journal of Neuroscience*, 32(6):2182–2190, 2012.

[112] Dinesh Jayaraman, Frederik Ebert, Alexei A Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. *International Conference on Learning Representations*, 2018.

[113] Sarah A Jelbert, Alex H Taylor, Lucy G Cheke, Nicola S Clayton, and Russell D Gray. Using the aesop's fable paradigm to investigate causal understanding of water displacement by New Caledonian crows. *PloS One*, 9(3):e92895, 2014.

[114] Philip N Johnson-Laird. Inference with Mental Models. In *The Oxford Handbook of Thinking and Reasoning*, pages 134–145. 2012.

[115] Keno Juechems and Christopher Summerfield. Where does value come from? Preprint, 2019.

[116] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011.

[117] Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current Opinions in Neurobiology*, 9(6):718–727, 1999.

[118] Rachel Keen. The development of problem solving in young children: A critical cognitive skill. *Annual review of psychology*, 62:1–21, 2011.

[119] Rachel Keen, Mei-Hua Lee, and Karen Adolph. Planning an action: A developmental progression in tool use. *Ecological Psychology*, 26(1-2):98–108, 2014.

[120] Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *PNAS*, 105(31):10687–10692, 2008.

[121] Daniel Kersten, Pascal Mamassian, and Alan Yuille. Object Perception as Bayesian Inference. *Annual Review of Psychology*, 55(1):271–304, 2004.

[122] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems*, pages 6348–6358, 2017.

[123] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[124] Wolfgang Kohler. *The mentality of apes*. Vintage Press, New York, 1927.

[125] Maria Kozhevnikov and Mary Hegarty. Impetus beliefs as default heuristics: dissociation between explicit and implicit knowledge about motion. *Psychonomic Bulletin & Review*, 8(3):439–53, sep 2001.

[126] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *arXiv preprint arXiv:1907.03146*, 2019.

[127] James Kubricht, Yixin Zhu, Chenfanfu Jiang, Demetri Terzopoulos, Song-Chun Zhu, and Hongjing Lu. Consistent Probabilistic Simulation Underlying Human Judgment in Substance Dynamics. In *Proceedings of the Annual Conference of the Cognitive Science Society*, 2017.

[128] Benjamin Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29(3):289–338, 1986.

[129] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[130] Scott Lembcke. Chipmunk 2d Physics Engine, 2013.

[131] Julia A Leonard, Yuna Lee, and Laura E Schulz. Infants make more attempts to achieve a goal when they see adults persist. *Science*, 357(6357):1290–1294, 2017.

[132] Yuan Chang Leong, Angela Radulescu, Reka Daniel, Vivian DeWoskin, and Yael Niv. Dynamic interaction between reinforcement learning and attention in multidimensional environments. *Neuron*, 93(2):451–463, 2017.

[133] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning*, 2016.

[134] Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2606–2613, 2017.

[135] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

[136] Tania Lombrozo. "Learning by thinking" in science and in everyday life. In *The Scientific Imagination*. Oxford University Press, New York, NY, 2018.

[137] Joao Loula, Kelsey Allen, Tom Silver, and Josh Tenenbaum. Learning constraint-based planning models from demonstrations. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[138] Joao Loula, Kelsey Allen, and Josh Tenenbaum. A task and motion approach to the development of planning. *Proceedings of the Cognitive Science Society*, 2020.

[139] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3684–3691. IEEE, 2014.

[140] Lydia V Luncz, Tiago Falótico, Alejandra Pascual-Garrido, Clara Corat, Hannah Mosley, and Michael Haslam. Wild capuchin monkeys adjust stone tools according to changing nut properties. *Sci Rep*, 6:33089, 2016.

[141] Roni O Maimon-Mor, Heidi Johansen-Berg, and Tamar R Makin. Peri-hand space representation in the absence of a hand–evidence from congenital one-handers. *Cortex; a journal devoted to the study of the nervous system and behavior*, 95:169, 2017.

[142] Roni O Maimon-Mor, Hunter R Schone, Rani Moran, Peter Brugger, and Tamar R Makin. Motor control drives visual bodily judgements. *Cognition*, 196:104120, 2020.

[143] Tamar R Makin, Meytal Wilf, Isabella Schwartz, and Ehud Zohary. Amputees “neglect” the space near their missing hand. *Psychological science*, 21(1):55–57, 2010.

[144] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[145] Alex Martin, Cheri L Wiggs, Leslie G Ungerleider, and James V Haxby. Neural correlates of category-specific knowledge. *Nature*, 379(6566):649–652, 1996.

[146] Michael E McCarty, Rachel K Clifton, and Roberta R Collard. Problem solving in infancy: The emergence of an action plan. *Developmental psychology*, 35(4):1091, 1999.

[147] Michael McCloskey. Intuitive physics. *Scientific American*, 248(4):122–130, 1983.

[148] Michael McCloskey and Deborah Kohl. Naive physics: the curvilinear impetus principle and its role in interactions with moving objects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9(1):146–156, 1983.

[149] Michael McCloskey, Allyson Washburn, and Linda Felch. Intuitive physics: the straight-down belief and its origin. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9(4):636, 1983.

[150] Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

[151] Douglas L Medin, Robert L Goldstone, and Dedre Gentner. Respects for similarity. *Psychological review*, 100(2):254, 1993.

[152] Alex Mesoudi, Lei Chang, Keelin Murray, and Hui Jing Lu. Higher frequency of social learning in china than in the west shows cultural variation in the dynamics of cultural evolution. *Proceedings of the Royal Society B: Biological Sciences*, 282(1798):20142209, 2015.

[153] Alex Mesoudi and Alex Thornton. What is cumulative cultural evolution? *Proceedings of the Royal Society B: Biological Sciences*, 285(1880):20180712, 2018.

[154] Christina M Metevier. Tool-using in rhesus monkeys and 36-month-old children: acquisition, comprehension, and individual differences, 2006.

[155] Tom Michael Mitchell. Version spaces: an approach to concept learning. Technical report, 1978.

[156] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

[157] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[158] Ida Momennejad. Learning structures: Predictive representations, replay, and generalization. *Current Opinion in Behavioral Sciences*, 32:155–166, 2020.

[159] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. "What happens if...": Learning to Predict the Effect of Forces in Images. In *European Conference on Computer Vision*, 2016.

[160] Allen Newell, J C Shaw, and Herbert A Simon. Elements of a Theory of Human Problem Solving. *Psychological Review*, 65(3), 1958.

[161] Allen Newell and Herbert A. Simon. *Human problem solving*. Human problem solving. Prentice-Hall, Oxford, England, 1972.

[162] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

[163] Yael Niv, Reka Daniel, Andra Geana, Samuel J. Gershman, Yuan Chang Leong, Angela Radulescu, and Robert C. Wilson. Reinforcement learning in multidimensional environments relies on attention mechanisms. *J. Neurosci*, 35:8145–8157, 2015.

[164] Robert M Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*, 115(1):39, 1986.

[165] Guy A. Orban and Fausto Caruana. The neural basis of human tool use. *Front Psychol*, 5, 2014.

[166] François Osiurak and Arnaud Badets. Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological review*, 123(5):534, 2016.

[167] François Osiurak, Christophe Jarry, and Didier Le Gall. Grasping the affordances, understanding the reasoning: toward a dialectical theory of human tool use. *Psychological review*, 117(2):517, 2010.

[168] François Osiurak and Emanuelle Reynaud. The elephant in the room: What matters cognitively in cumulative technological culture. *Behavioral and Brain Sciences*, pages 1–57, 2020.

[169] François Osiurak. What Neuropsychology Tells us About Human Tool Use? The Four Constraints Theory (4ct): Mechanics, Space, Time, and Effort. *Neuropsychology Review*, 24(2):88–115, 2014.

[170] François Osiurak, Christophe Jarry, Mathieu Lesourd, Josselin Baumard, and Didier Le Gall. Mechanical problem-solving strategies in left-brain damaged patients and apraxia of tool use. *Neuropsychologia*, 51(10):1964–1972, 2013.

[171] Pierre-Yves Oudeyer and Linda B Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2):492–502, 2016.

[172] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[173] Derek C. Penn and Daniel J. Povinelli. Causal Cognition in Human and Nonhuman Animals: A Comparative, Critical Review. *Annual Review of Psychology*, 58(1):97–118, 2007.

[174] Madison Pesowski, Alyssa Quy, Michelle Lee, and Adena Schachner. Children use inverse planning to detect social transmission in design of artifacts. 2020.

[175] Daniel J Povinelli. *Folk physics for apes: The chimpanzee's theory of how the world works*, volume 7. 2000.

[176] Dennis R. Proffitt and David L. Gilden. Understanding natural dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 15(2):384–393, 1989.

[177] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[178] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[179] John C Raven and JH Court. *Standard Progressive Matrices: Sets A, B, C, D & E.* HK Lewis London, 1958.

[180] Bruce Rawlings and Cristine H Legare. Toddlers, tools, and tech: The cognitive ontogenesis of innovation. *Trends in Cognitive Sciences*, 2020.

[181] A David Redish, Steve Jensen, Adam Johnson, and Zeb Kurth-Nelson. Reconciling reinforcement learning models with behavioral extinction and renewal: implications for addiction, relapse, and problem gambling. *Psychological review*, 114(3):784, 2007.

[182] Stephen K Reed. Pattern recognition and categorization. *Cognitive psychology*, 3(3):382–407, 1972.

[183] Adam N. Sanborn, Vikash K. Mansinghka, and Thomas L. Griffiths. Reconciling intuitive physics and Newtonian mechanics for colliding objects. *Psychol Rev*, 120(2):411–437, 2013.

[184] Takao Sasaki and Dora Biro. Cumulative culture can emerge from collective intelligence in animal groups. *Nature communications*, 8(1):1–6, 2017.

[185] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[186] Anna C Schapiro, Timothy T Rogers, Natalia I Cordova, Nicholas B Turk-Browne, and Matthew M Botvinick. Neural representations of events arise from temporal community structure. *Nature neuroscience*, 16(4):486–492, 2013.

[187] Jürgen Schmidhuber, Jieyu Zhao, and Nicol N Schraudolph. Reinforcement learning with self-modifying policies. In *Learning to learn*, pages 293–309. Springer, 1998.

[188] Daniel P Schofield, William C McGrew, Akiko Takahashi, and Satoshi Hirata. Cumulative culture in nonhumans: overlooked findings from japanese monkeys? *Primates*, 59(2):113–122, 2018.

[189] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[190] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[191] Daniel L Schwartz and John B Black. Shuttling between depictive models and abstract rules: Induction and fallback. *Cognitive Science*, 20(4):457–497, 1996.

[192] Sarah Schwettmann, Joshua B Tenenbaum, and Nancy Kanwisher. Invariant representations of mass in the human brain. *eLife*, 8:e46619, 2019.

[193] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[194] Robert Shaw. Ecological psychology: The consequence of a commitment to realism. Erlbaum, 1982.

[195] Robert Shaw. The agent-environment interface: Simon's indirect or gibson's direct coupling? *Ecological Psychology*, 15(1):37–106, 2003.

[196] Roger N. Shepard. Multidimensional scaling, tree-fitting, and clustering. *Science*, 210(4468):390–398, 1980.

[197] Robert W Shumaker, Kristina R Walkup, and Benjamin B Beck. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011.

[198] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, and Koray Kavukcuoglu. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7585):484–489, 2016.

[199] Tom Silver, Kelsey R Allen, Alex K Lew, Leslie Pack Kaelbling, and Josh Tenenbaum. Few-shot bayesian imitation learning with logical program policies. In *AAAI*, pages 10251–10258, 2020.

[200] Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 816–823, 2005.

[201] Kevin A Smith, Peter W Battaglia, and Edward Vul. Consistent physics underlying ballistic motion prediction. In *Proceedings of the 35th Annual Meeting of the Cognitive Science Society*, 2013.

[202] Kevin A Smith and Edward Vul. Sources of uncertainty in intuitive physics. *Topics in Cognitive Science*, 5(1):185–199, 2013.

[203] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017.

[204] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007.

[205] John P Spencer, Linda B Smith, and Esther Thelen. Tests of a dynamic systems account of the a-not-b error: The influence of prior experience on the spatial memory abilities of two-year-olds. *Child development*, 72(5):1327–1346, 2001.

[206] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[207] James W. Stigler. "Mental abacus": The effect of abacus training on Chinese children's mental calculation. *Cognitive Psychology*, 16(2):145–176, 1984.

[208] Dietrich Stout and Thierry Chaminade. Stone tools, language and the brain in human evolution. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1585):75–87, 2012.

[209] Erik M Summerside, Reza Shadmehr, and Alaa A Ahmed. Vigor of reaching movements: reward discounts the cost of effort. *Journal of neurophysiology*, 119(6):2347–2357, 2018.

[210] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pages 216–224. Elsevier, 1990.

[211] Richard S. Sutton. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bull.*, 2(4):160–163, 1991.

[212] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[213] Serge Thill, Daniele Caligiore, Anna M. Borghi, Tom Ziemke, and Gianluca Baldassarre. Theories and computational models of affordance and mirror systems: An integrative review. *Neuroscience & Biobehavioral Reviews*, 37(3):491–521, 2013.

[214] Bill Thompson and Tom Griffiths. Inductive biases constrain cumulative cultural evolution. In *Proceedings of the Cognitive Science Society*, pages 1111–1117, 2019.

[215] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[216] Michael Tomasello. *The Cultural Origins of Human Cognition*. Harvard University Press, 1999. Google-Books-ID: hRFfCgAAQBAJ.

[217] Momchil Tomov, Eric Schulz, and Samuel J Gershman. Multi-task reinforcement learning in humans. *bioRxiv*, page 815332, 2019.

[218] Marc Toussaint, Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems*, 2018.

[219] Ernő Téglás, Edward Vul, Vittorio Girotto, Michel Gonzalez, Joshua B. Tenenbaum, and Luca L. Bonatti. Pure Reasoning in 12-Month-Old Infants as Probabilistic Inference. *Science*, 332(6033):1054–1059, 2011.

[220] Krist Vaesen. The cognitive bases of human tool use. *Behav Brain Sci*, 35(4):203–218, 2012.

[221] Michiel van Elk, Hein van Schie, and Harold Bekkering. Action semantics: A unifying conceptual framework for the selective use of multimodal and modality-specific object knowledge. *Physics of life reviews*, 11(2):220–250, 2014.

[222] Gilles Vannuscorps and Alfonso Caramazza. Typical action perception and interpretation without motor simulation. *Proceedings of the National Academy of Sciences*, 113(1):86–91, 2016.

[223] Gilles Vannuscorps, Agnesa Pillon, and Michael Andres. Effect of biomechanical constraints in the hand laterality judgment task: where does it come from? *Frontiers in Human Neuroscience*, 6:299, 2012.

[224] Wolf Vanpaemel, Gerrit Storms, and Bart Ons. A varying abstraction model for categorization. In *Proceedings of the Annual Conference of the Cognitive Science Society*, volume 27, pages 2277–2282, 2005.

[225] Elisabetta Visalberghi and Luca Limongelli. Lack of comprehension of cause and effect relations in tool-using capuchin monkeys (cebus apella). *Journal of Comparative Psychology*, 108(1):15, 1994.

[226] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. NerveNet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

[227] Christine E Watson and Laurel J Buxbaum. Uncovering the architecture of action semantics. *Journal of Experimental Psychology: Human Perception and Performance*, 40(5):1832, 2014.

[228] Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, David Silver, and Daan Wierstra. Imagination-Augmented Agents for Deep Reinforcement Learning. In *NIPS 2017*, 2017.

[229] Sam Wenke, Dan Saunders, Mike Qiu, and Jim Fleming. Reasoning and generalization in rl: A tool use perspective. *arXiv preprint arXiv:1907.02050*, 2019.

[230] Peter A. White. The experience of force: The role of haptic experience of forces in visual perception of object motion and interactions, mental simulation, and motion-related judgments. *Psychological Bulletin*, 138(4):589–615, 2012.

[231] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[232] Margaret Wilson. Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4):625–636, 2002.

[233] Joanna H Wimpenny, Alex AS Weir, Lisa Clayton, Christian Rutz, and Alex Kacelnik. Cognitive processes associated with sequential tool use in new caledonian crows. *PLoS One*, 4(8):e6471, 2009.

[234] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

[235] Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

[236] Jiajun Wu, Ilker Yildirim, Joseph J. Lim, William T. Freeman, and Joshua B. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems*, 2015.

[237] Liyu Xia and Anne G.E. Collins. Temporal and state abstractions for efficient learning, transfer and composition in humans. *bioRxiv*, 2020.

[238] Victoria Xia, Zi Wang, Kelsey Allen, Tom Silver, and Leslie Pack Kaelbling. Learning sparse relational transition models. *International Conference on Learning Representations*, 2018.

[239] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *Robotics: Science and Systems*, 2019.

[240] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[241] Ilker Yildirim, Tobias Gerstenberg, Basil Saeed, Marc Toussaint, and Josh Tenenbaum. Physical problem solving: Joint planning with symbolic, geometric, and dynamic constraints. In *Proceedings of the Annual Conference of the Cognitive Science Society*, 2017.

[242] Ilker Yildirim and Robert A Jacobs. Transfer of object category knowledge across visual and haptic modalities: Experimental and computational studies. *Cognition*, 126(2):135–148, 2013.

[243] Ilker Yildirim, Basil Saeed, Grace Bennett-Pierre, Tobias Gerstenberg, Joshua Tenenbaum, and Hyowon Gweon. Explaining intuitive difficulty judgments by modeling physical effort and risk. In *Proceedings of the Annual Meeting of the Cognitive Science Society.*, 2019.

[244] Ilker Yildirim, Kevin A. Smith, Mario Belledonne, Jiajun Wu, and Joshua B. Tenenbaum. Neurocomputational Modeling of Human Physical Scene Understanding. In *Conference on Cognitive Computational Neuroscience*, 2018.

[245] Jan Zalasiewicz, Mark Williams, Alan Haywood, and Michael Ellis. The anthropocene: a new epoch of geological time?, 2011.