Artificial Intelligence 300 (2021) 103551

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Abstraction in data-sparse task transfer

Tesca Fitzgerald[a,*,1], Ashok Goel[b], Andrea Thomaz[c]

[a] Carnegie Mellon University, Pittsburgh, PA, USA
[b] Georgia Institute of Technology, Atlanta, GA, USA
[c] University of Texas at Austin, Austin, TX, USA

A R T I C L E I N F O

Article history:
Received 22 October 2020
Received in revised form 6 April 2021
Accepted 22 June 2021
Available online 29 June 2021

Keywords:
Cognitive robotics
Task transfer
Task representation
Interactive robot learning
Abstraction

A B S T R A C T

When a robot adapts a learned task for a novel environment, any changes to objects in the novel environment have an unknown effect on its task execution. For example, replacing an object in a pick-and-place task affects where the robot should target its actions, but does not necessarily affect the underlying action model. In contrast, replacing a tool that the robot will use to complete a task will effectively alter its end-effector pose with respect to the robot's base coordinate system, and thus the robot's motion must be replanned accordingly.

These examples highlight the relationship among (i) differences between the source and target environments, (ii) the level of abstraction at which a robot's task model should be represented to enable transfer to the target environment, and (iii) the information needed to ground the abstracted task representation in the target environment. In this article, we present a taxonomy of transfer problems based on this relationship. We also describe a knowledge representation called the Tiered Task Abstraction (TTA) and demonstrate its applicability to a variety of transfer problems in the taxonomy. Our experimental results indicate a trade-off between the generality and data requirements of a task representation, and reinforce the need for multiple transfer methods that operate at different levels of abstraction.

1. Introduction

Adaptability is an essential skill in human cognition, enabling us to draw from our extensive, life-long experiences with various objects and tasks in order to address novel problems. To date, robots do not have this kind of adaptability; yet, as our expectations of robots' interactive and assistive capacity grows, it will be increasingly important for them to adapt to unpredictable environments in a similar manner as humans.

While a robot can be pre-programmed for many tasks and their variations, specifying these behaviors would require tedious effort, and still would not adequately prepare a robot for every scenario it may encounter. This is due to the various dimensions along which the original ("source") and novel ("target") task may differ, such as changes in the task goals, task objects, manipulation tools, task constraints, and task dynamics.

Prior research in Machine Learning has enabled robots and virtual agents to learn a "task model" that accepts a representation of the problem state as input and outputs a series of actions that, when executed by the robot/agent, results in

* Corresponding author.
E-mail addresses: tesca@cmu.edu (T. Fitzgerald), goel@cc.gatech.edu (A. Goel), athomaz@ece.utexas.edu (A. Thomaz).
[1] Work completed while at Georgia Institute of Technology.

https://doi.org/10.1016/j.artint.2021.103551

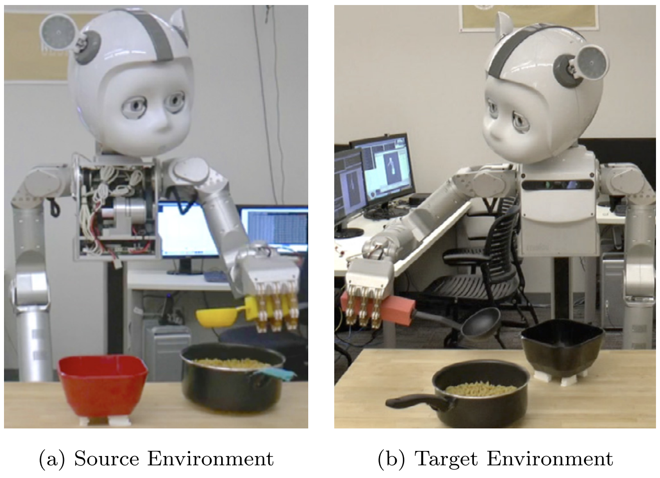


Fig. 1. The robot learns a scooping task in a known, *source* environment (a). At a later time, the robot aims to transfer its model of the scooping task to a novel, *target* environment (b) containing a new bowl, displaced objects, and a new scoop tool. Before attempting to complete the scooping task in the target environment, the robot must adapt its learned task model to accommodate these differences between the source and target environments.

successful completion of the task. As this task model is trained over more examples of the task, its ability to successfully generalize to unseen task variations typically increases as well. When applied to a physical robot, these examples may be derived by allowing the robot to continue to explore the task through trial and error, or by having a human teacher provide enough demonstrations of the task (via interactive task learning [1]) to cover the space of possible task variations. However, these demonstrations are not sufficient to span the full space of environment variations the robot may encounter, nor their effect on task completion. We consider a data-sparse paradigm instead, where the robot transfers a *specific* task model from a source environment in order to address a related target environment containing a new set of objects.

This problem requires an understanding of how task differences, abstraction, and transfer are related. We particularly focus on the effect of changes in the robot's *environment* on task execution (rather than adapting to changes in other aspects of the task, such as reward functions or task goals). Within the scope of addressing environment changes, there are several sources of novelty that are introduced by object changes or replacements. For example, changes in the location, dimensions, 3D shape, and/or affordances of a new object must all be addressed for the robot to transfer a task model successfully to the new environment. However, these changes will affect the task in different ways. For example, relocating an object will have a minor effect on task execution compared to replacing one tool object with another (in which case, the task adaptation depends on how the tool is used within the context of the task, e.g. Fig. 1). As a result, the type of novelty encountered in a target environment affects how a robot should address this novelty.

In this article, we first characterize the problem of task transfer in terms of **similarity** between the source and target environments. In doing so, **we present a taxonomy of transfer problems** that models the relationship between state space changes and information requirements for transfer; the difference between source and target problems dictates (i) the level of abstraction at which the task representation should be transferred and (ii) the dimensionality of the information needed to ground that representation for the target problem. Based on this taxonomy, **we define the Tiered Task Abstraction (TTA)**: a generalized task representation that implements these principles of task similarity and abstraction. We then implement this representation in a **case study demonstrating the TTA representation**'s effectiveness on a physical robot transferring two pick-and-place tasks to a variety of target environments. Finally, we conclude this article with a discussion of autonomous and collaborative methods for grounding an abstracted task representation in a new environment.

2. Related work

The aim of *transfer learning* is to use knowledge of a source task to improve the agent's performance in completing a related, target task. Improved performance may be measured according to metrics such as better initial performance in the target task, increased learning speed in the target task, or fewer training instances of the target task [2]. The difference between the robot's known, "source" data and the new, "target" problem directly affects the data and method used for transfer. Taylor and Stone provide a breakdown of transfer learning approaches for RL domains according to the *task differences* they are equipped to address [2]. In the remainder of this section, we summarize research addressing (i) low-level task differences affecting the robot's action model parameterization and (ii) high-level task differences affecting task specification, organization, and/or constraints. Note that our usage of the term "task model" is derived from the Machine Learning literature, and is distinct from the definition of "task models" used in the HCI literature, in which a task model represents a user's goals, activities, and roles when using a computer interface [3].

2.1. Action-level transfer in robotics

Learning from Demonstration (LfD) is a well-researched process in which an AI agent attempts to learn a policy π that models one or more examples ("demonstrations") provided by a teacher (potentially an oracle or a human teacher) [4,5]. Each example consists of an input *state* s and output *action* a that the agent's policy should return in that state such that $\pi(s) = a$. In the context of robot LfD, a demonstration is typically a series of state-action pairs representing the *trajectory* that the teacher demonstrates to complete a task, e.g. $d = \{< s_0, a_0 >, < s_1, a_1 >, \cdots < s_n, a_n >\}$. The state and action representations encoded in this demonstration are dependent on the type of interaction used to provide demonstrations. In this paper, we focus on the use of kinesthetic demonstrations, where a human teacher provides demonstrations by moving the robot's own arm to complete the task [6]. As a result, the states and actions recorded during the demonstration are known to be attainable by the robot itself at a later time.

After the robot has recorded the interactive demonstration, it must model the recorded data in order to produce a similar action at a later time. The model used to represent this data directly affects its generalizability. Dynamic Movement Primitives (DMPs) [7] provide a dynamical system approach to task generalization, in which the demonstration trajectory is represented as a perturbed spring-damper system. Pastor et al. [8] demonstrate how this model enables generalization to spatial perturbations of the learned task. While DMPs enable generalization from a single task demonstration, it does not represent the relationship between non-spatial changes in the robot's environment (such as changes in object features) and the corresponding model adaptation. Gaussian Mixture Models (GMMs) can be used to represent variations across multiple demonstrations at their corresponding datapoints, with this variance measured with respect to a feature vector defined by observations of the robot's environment [9]. However, GMMs are dependent on a training dataset containing a similar distribution of parameters as those of the target environments.

More recent work has leveraged training data across multiple tasks in order to reduce the training data needed to generalize to a new task (or new task variation). Bruno, Calinon, & Caldwell [10] present a method for learning a task model that is separate from parameters of the task, which can be determined according to the location of the robot's end-effector at the beginning and end of a primitive action [11]. Recent work in one/few-shot learning [12–15] has focused on learning a non-linear relationship between environment features and their effect on the task parameters across multiple tasks, such that this model can be quickly tuned for a new task. However, these approaches rely on training data being readily available, and thus is not suited for the sparsity of data collected in an interactive setting. Additionally, the task goals may affect how the differences between the source and target tasks and/or environments effect the task completion. We address the problem of a robot that has *not yet* been able to explore the effect of these differences on the task goals, and thus needs to learn a task-specific relationship between perceptual changes and the robot's action models.

2.2. Task-level transfer in cognitive systems

Rather than attempt to learn a single model that is generalizable across multiple problems, we now consider cognitively-inspired approaches that aim to identify the relationships between *specific* problem-solution pairs. As a result, these approaches do not typically involve training a generative model over a set of data, but instead perform an analysis over the relationships embedded within and between problem-solution pairs. We now consider how these approaches apply to the problem of task transfer.

Analogical reasoning is a cognitively-inspired methodology for applying past experiences to a new, unfamiliar problem [16–18]. The source cases and target problems in analogical reasoning lie on a *similarity spectrum* [19]. At one end of the spectrum, the target case is identical to a source case, and thus can be transferred to the target problem without any adaptation. At the other end of the spectrum, the target problem is completely dissimilar to all source cases available in the case memory and thus cannot be addressed via transfer. In between the two extremes, transfer entails problem abstraction where the level of abstraction may depend on the degree of similarity between the source and target problems [20].

Case-Based Reasoning (CBR) can be applied to problems of analogy in which similar problems are assumed to have similar solutions [16]. Experiences are stored individually as source cases in a *source case memory*, and are retrieved and adapted to address an unfamiliar problem (the *target* problem) by (1) retrieving the most relevant source case from memory, (2) creating a mapping that outlines the task differences between the source and target cases, (3) adapting and deploying the source case according to the mapping, (4) evaluating the transferred case's performance, and (5) saving the revised case as a new source case for later usage.

While case-based reasoning has been applied to some problems in robotics and perceptual domains [21–23], it is typically employed at the strategic level and thus relies on an abstracted representation of the task or goal. This produces an additional challenge when applied to the context of a robot's high-dimensional representations of action and perception.

2.3. Grounding task representations in robot actions and perception

Task *grounding* is an alternative to task generalization, in which the objective is to execute a known task model in a new, specific environment (e.g. "task recipes" [24,25] or language-based task representations [26]). This often necessitates a mapping between the objects referenced in the task model and their counterparts in the target environment. The robot may infer the object mapping based on similarity between the objects' features [27], but this relies on hand-coded heuristics for object similarity that may be task- or domain-specific [28]. While these approaches are useful for applying a task recipe to a specific environment, they rely on the robot having an abstracted task representation in the first place. Task recipes are designed to be generalizable across robots and environments; while a human with knowledge of the task can specify a task recipe by hand, a robot is unable to do so directly from task demonstrations.

2.4. Summary of related works

The high-dimensionality of robot perception and action presents a challenge when transferring a learned task model to a new environment. Demonstrations alone are insufficient to enable a robot to generalize to unforeseen task differences as they do not encode the relationship between environment changes and task model modifications.

While analogical reasoning and CBR provide useful frameworks for selecting and adapting source cases for transfer based on high-level task differences, they are best suited for domains in which the action and state space representations are *already* abstracted such that the relationship between the source and target tasks is apparent.

Task grounding provides one tool for bridging this gap between low-level and high-level task representations for transfer. In this article, we focus on learning a mapping between the robot's source and target environments *without* relying on hand-coded mapping heuristics. To do so, we aim to leverage the grounded, contextual knowledge imparted by demonstrations, while also maximizing the robot's autonomy by limiting the amount of interactive assistance required by the robot. Furthermore, while prior work has demonstrated how a robot can ground a specific abstraction of a task representation, we aim to propose a task representation that can be abstracted at *multiple levels* in order to adapt to a wider range of transfer problems. This requires an understanding of (i) the data that is needed to ground the task representation at each level of abstraction and (ii) the modes of interaction that the robot may use to obtain this data. We next analyze the relationship between these two factors, as well as how they are dependent on the similarity between source and target environments.

3. Taxonomy: categorizing task differences

We refer to a *task* as a sequence of object-oriented task steps or *skills*, each consisting of their own action model and performed in series to achieve a goal. As an example, a *cup-pouring* task would consist of three action models: (i) grasping the cup, (ii) lifting the cup, and (iii) tipping the cup, each defined with respect to the cup's pose in the robot's environment. This definition results in three key elements of a task representation: the robot's state with respect to its environment (e.g. objects), the action model comprising each skill, and the goal that is achieved by its execution. These correspond to the state space, action space, and goal/rewards commonly used to define a Markov Decision Process or other task-planning problem. We next consider how changes to any of these three defining task elements affect the robot's execution of the task. In later sections of this article, we address **only changes in the state-space** through our approach.

3.1. Goal space changes

A representation of the task goal(s) may be used to guide transfer by defining the preconditions, postconditions, or constraints that must be met to successfully complete the task. Reasoning directly over changes in the robot's goal state would enable the robot to reuse previously-learned task models to fulfill new goals, such as inverting a model for an assembly task so that it can be re-purposed for a disassembly task. As introduced in the previous section, analogical reasoning is a cognitively-inspired approach that is well suited to adapt to changes in the goal representation. Prior work in this research area operates over a set of *source cases*, each containing an instance of a problem-solution pair, stored in a *source case memory*. An unfamiliar problem is then addressed using the following methodology. As new problems are viewed, the single most similar observed case is pulled from the source case memory. The retrieved case is then compared to the new, *target case*, and a mapping is derived that contains the differences between the two. Using this mapping, the retrieved source case solution is then adapted to address the differences between the two cases. The adapted solution is then deployed in the context of the target case.

A central goal of analogical reasoning is identifying the common relationships between problem-solution pairs. This relies on having a representation of both the problem and solution that is abstract enough for these relationships to be identified. A task goal that is defined symbolically (e.g. as a set of pre-/post-conditions) could support analogical reasoning; however, this requires that the symbolic representation be learned or otherwise defined (e.g. by a human teacher or a dataset).

Prior work in Learning from Demonstration has shown how a robot may learn a goal model through demonstrations, resulting in a representation such as a probabilistic goal model [29], a series of task constraints [30], a goal descriptor [31], or an abstracted skill tree [32]. These methods require multiple demonstrations of various successful and unsuccessful goal states, and/or goal specifications by a human teacher. Rather than assume that the robot has received a sufficiently large set of demonstrations or that the goal has been manually specified by a human, we presume that the robot does not have access to a goal model to facilitate transfer of its learned action models to a target problem.

3.2. Action space changes

Action space changes occur when transferring a learned task model to a robot with different kinematics, constraints, and/or output modalities. Additionally, new kinematic constraints may be introduced or removed, also resulting in a change in the robot's action space. Transfer learning methods have been used to address these task differences with the goal of transferring knowledge gained in the source domain to improve an agent's performance and/or learning speed in the target domain [2,33]. In relation to skill learning, this may involve transferring an action policy that is learned in one domain such that it can be used to reduce the time required to repeat or relearn the skill in a second domain. Transfer via inter-task mapping [33] enables transfer learning for reinforcement learning agents with similar goals but different action spaces in domains such as RoboCup Keepaway. Within the context of a single robot that operates in human environments, we do not expect that the robot's action space will change, and thus do not address this category of transfer problems.

3.3. *State space changes*

In this article, we only address transfer problems that result from changes in the robot's state space, as opposed to changes in the robot's goal or action spaces. Furthermore, we limit our scope to state space changes that are known to the robot, either through the robot's own perception or through information provided to it (e.g. by a human teacher's description of the task). The robot's state space is typically defined in terms of the relationship between the robot's kinematic state and its environment. Regardless of the exact state space specification used, the environmental variations that are common in human environments are likely to be reflected in the robot's state space as well. We categorize these variations as follows:

- *Perceptual* changes in which the robot's environment appears different while remaining structurally and/or functionally the same. E.g. changes in lighting or in the appearance of an object.
- *Structural* changes in which the relationship between objects within the robot's environment is altered. E.g. when objects are moved around the scene.
- *Functional* changes that affect the relationship between the robot and its environment. E.g. the introduction of obstacles that constrain the robot's motion, or the use of an object as a new end-effector.

Note that some changes to the robot's state space may result in multiple effects. For example, moving a block to another part of the scene constitutes a structural change, but may also have the effect of occluding the robot's view of a second object, thus creating a perceptual change in the robot's representation of that second object. Prior work has addressed the problem of structural changes in the robot's state space. Pastor et al. [8] describe an approach to learning a series of primitive skill models which comprise complex tasks. A Dynamic Movement Primitive (DMP) is trained over a demonstration by perturbing a linear spring-damper system according to the velocity and acceleration of the robot's end-effector at each time step [7,8]. By integrating over the DMP, a trajectory can then be generated that begins at the end-effector's initial position and ends at a specified end point location. Thus, after training a DMP, the only parameter required to execute the skill is the desired end point location. By parameterizing the end point location of each DMP skill model according to object locations, the overall task can be generalized to accommodate new object configurations.

3.4. *Relationship between abstraction and similarity*

Fig. 2b illustrates how these state space changes may be expressed. A task demonstrated in a source environment (e.g. a scooping task performed in the environment shown in Fig. 2a) can be directly reused in a target environment which either (i) does not require modification of the learned task (image 1 in Fig. 2b), or (ii) has a known parameterization according to the target environment. For example, image 2 in Fig. 2b demonstrates a target environment containing a *structural* change: repositioned objects. If the robot has learned a task model that is parameterized with respect to the location of objects in the scene, it can address this target environment by re-evaluating its parameter values according to the objects' new positions.

When addressing a target environment that exhibits *perceptual* changes (image 3 in Fig. 2b), the objects in that environment are unfamiliar but serve the same purpose as objects in the source environment and do not need to be manipulated differently. This problem can be addressed by identifying a mapping between objects in the source and target environments. This mapping can be used to ground the task parameters in the target environment's feature values rather than the source environment. For task models that are defined with respect to specific objects in the robot's environment, these object references must be updated when one or more objects are replaced in the robot's environment. This requires a similarity metric that may be used to determine a mapping between objects in the source and target environments.

Image 4 in Fig. 2b differs from the source (Fig. 2a) in that objects are: (i) displaced, (ii) replaced, and now (iii) *constrained*. This constraint is a result of the role that the scoop object plays in the task; some object replacements may have additional effects on task transfer, such as when replacing an object that is used as a *tool* to complete the task. The shape of a tool alters its effect on its environment [34], and thus a tool replacement may necessitate a change in the manipulation of that tool in order to achieve the same task goal [35]. Since the scoop is used as an end-effector during a scooping task, the robot's actions must now be constrained such that its end-effector remains higher above the table in order to complete the task successfully with the larger scoop. This presents a *functional* change, since the relationship between the robot and its environment has been changed due to the tool replacement. Since the task model was trained with respect to the source environment (and as a result, the source scoop tool), its output must be mapped to the corresponding actions using the target scoop.

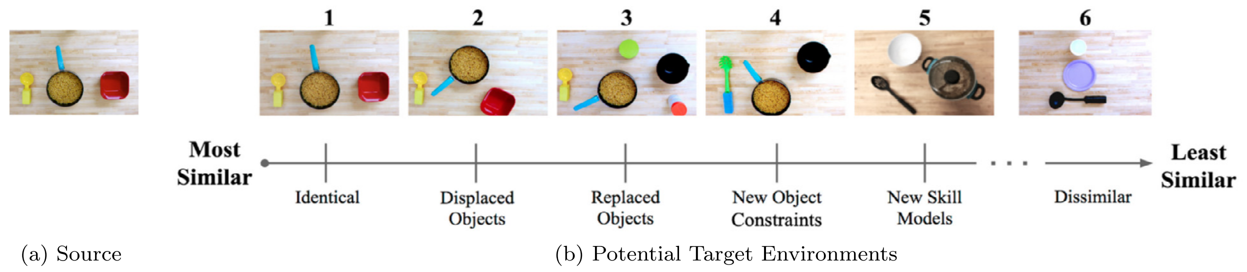


Fig. 2. Spectrum of Similarity Between Source and Target Environments.

Image 5 in Fig. 2b differs from the source in similar respects, with one additional difference: an extra step is needed in order to lift the lid off the pasta pot prior to scooping the pasta. As a result, the original skill models learned in the source cannot be directly transferred. In addition to deriving an object mapping and action mapping as in the previous transfer problems, this target environment also requires that the robot derive or learn a new action model to account for the missing step.

These task differences illustrate a *spectrum* of similarity between the source and target; at one end of the spectrum, the source and target differ in ways that have a small effect on the robot's execution of the task, such as object configurations. At the other end of the spectrum, they contain more differences, until finally (as in image 6 in Fig. 2b), the target environment cannot be addressed via transfer. While we have highlighted discrete levels of similarity in this spectrum, we do not claim this to be an exhaustive categorization of transfer problems. In prior work, we have also explored how a robot may need to exhibit creativity in order to address more dissimilar target environments [36]. In summary, Fig. 2 illustrates that without further information about the task as it pertains to the target environments, task transfer methods are limited to addressing a narrow set of transfer problems: those in which the target environment does not require novel behavior or reasoning to address.

4. Representation: Tiered Task Abstraction (TTA)

The previous section described how task differences affect the robot's task completion differently, and thus require different information in order to transfer the learned task model to a new environment. We propose the Tiered Task Abstraction (TTA) representation to *address transfer problems consisting of state space changes* such as those shown in Figure 2. We aim for this representation to reflect the relationship between (1) changes in the state space and (2) their effect on the task transfer. While other categories of transfer problems, such as changes in the robot's action or goal spaces, are likely to be encountered by robots that operate in human environments, these problem categories are outside the scope of the TTA representation.

Overall, the Tiered Task Abstraction representation consists of four elements: an action model **a**, parameterization function **p**, feature selector **f**, and feature values E. These four elements represent a task demonstration as a single action model as follows:

$$\mathbf{a_i}(\mathbf{p_0}(\mathbf{f_0}(E)), \mathbf{p_1}(\mathbf{f_1}(E)), \ldots \mathbf{p_m}(\mathbf{f_m}(E)) \tag{1}$$

Or as a *series* of action models as follows:

$$\mathbf{a_0}(\mathbf{p_0}(\mathbf{f_0}(E))), \ldots, \mathbf{a_m}(\mathbf{p_m}(\mathbf{f_m}(E))) \tag{2}$$

where:

- $\mathbf{a_i}(\mathbf{p})$ is an **action model** that is parameterized by p and outputs a trajectory consisting of a series of poses for the robot to execute. Depending on the complexity of the task, multiple action models may be needed for the robot to complete the task, as formulated in Equation (2). In this formulation, each action model may represent a single sub-sequence of the task, such as defining one action model to use a cup to scoop water from a bowl and a second action model to then pour the cup in the next task step. Prior work in robot LfD has defined many possible implementations for the action model, such as Gaussian Mixture Models [9,37], Dynamic Movement Primitives [7,11], or neural networks [14,38]. Regardless of how the action model is implemented, we presume that it is trained over one or more task demonstrations and is adapted for a particular task instance according to some parameter vector p.
- $\mathbf{p_i}(\mathbf{f})$ is a **parameterization function** for the action model a. The parameters returned by this function serve as a task-relevant encoding of the input features f. A common example of the parameters that should be returned by this function include the goal state or dynamics constraints that are provided as input to the action model. As a result, the implementation of this parameterization function is dependent on the action model that is used.
- $\mathbf{f_i}(\mathbf{E})$ serves as a **feature selector** that returns task-relevant feature representation of the robot's **input space** E. This function also serves to extract a state space representation from a potentially high-dimensional perceptual input, such as that derived from an RGBD sensor. (See Fig. 3.)

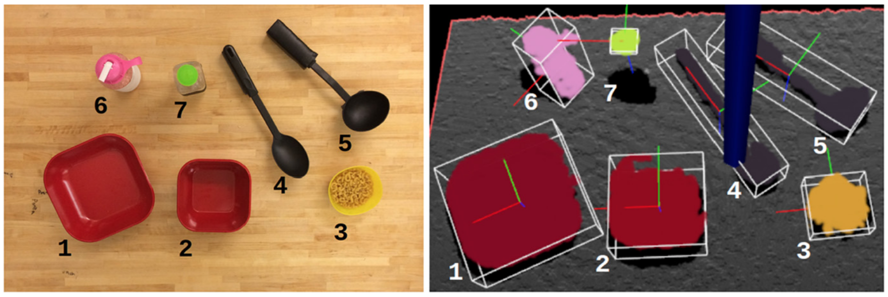
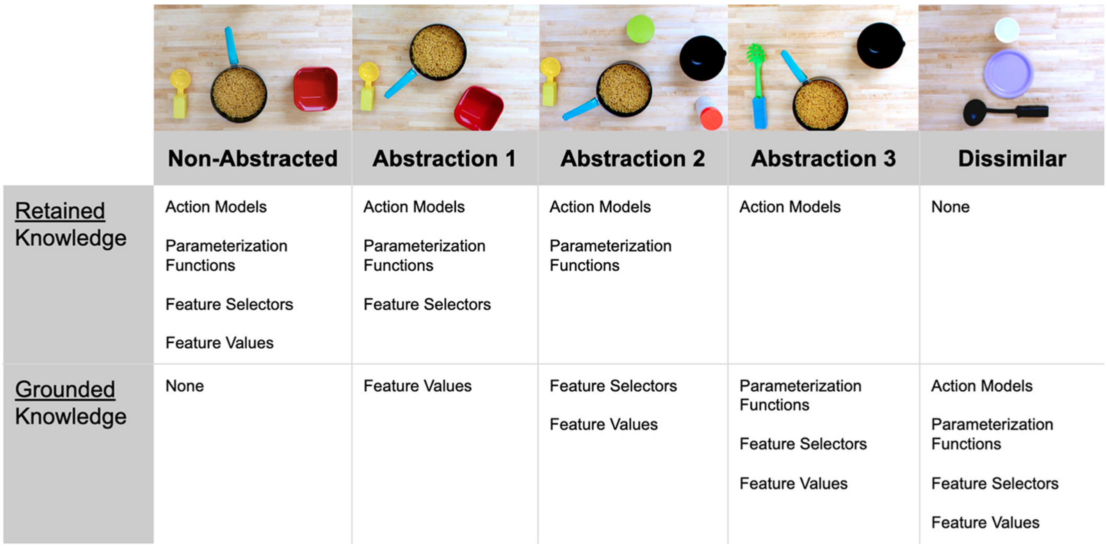


Fig. 3. An overhead view of a table-top environment (left) and the segmented point cloud representation (right).



	Non-Abstracted	Abstraction 1	Abstraction 2	Abstraction 3	Dissimilar
Retained Knowledge	Action Models Parameterization Functions Feature Selectors Feature Values	Action Models Parameterization Functions Feature Selectors	Action Models Parameterization Functions	Action Models	None
Grounded Knowledge	None	Feature Values	Feature Selectors Feature Values	Parameterization Functions Feature Selectors Feature Values	Action Models Parameterization Functions Feature Selectors Feature Values

Fig. 4. The grounding requirements for each level of abstraction.

This definition of each element in the TTA representation is intentionally conceptual, as the specific implementation, input, and output of each of these functions is dependent on the robot and the domain in which it is deployed. In Section 5, we describe one such implementation of this representation in a specific use case. Regardless of the specific function implementations, we make two key assumptions about the robot's input and output space. First, we assume that these functions are trained a priori on *trajectory* data demonstrated by a human teacher and consisting of k robot poses recorded throughout the demonstration. Second, we assume that the robot's state space is encapsulated by the feature values E. In our implementation described in Section 5, this input consists solely of the robot's perceptual input derived from an RGBD sensor. However, in another domain, it is possible that the state may consist of information that cannot be perceived by the robot, and thus the input E may need to be derived from additional sources.

The defining characteristic of the TTA representation is that each element is parameterized by the next. By omitting one or more elements from the task representation, the resulting representation is one that is *abstracted*. In doing so, TTA enables a task to be represented at the level of abstraction that is common to both the source and target environments. Fig. 4 defines three abstractions of this representation. However, once a representation is abstracted, it must be *grounded* in the target environment in order to produce an output that is executable by the robot. In an embodied system, grounding refers to parameterizing a representation based on perception in the physical world. A representation is *grounded* in a target environment when all of its elements (action model, parameterization function, feature selector, and feature values) are present and defined based on information derived in the target environment (either by perception or interaction in the target environment). If the representation cannot be fully grounded in a target environment, the robot may need to re-learn the task within the context of the target environment. We summarize the grounding requirements of each abstraction level in Fig. 4.

5. Case study: transferring a task model at multiple abstraction levels

We evaluate whether the Tiered Task Abstraction reflects the relationship between task differences and the resulting data requirements to enable task transfer. To do this, we represent the same task model at three abstraction level and test

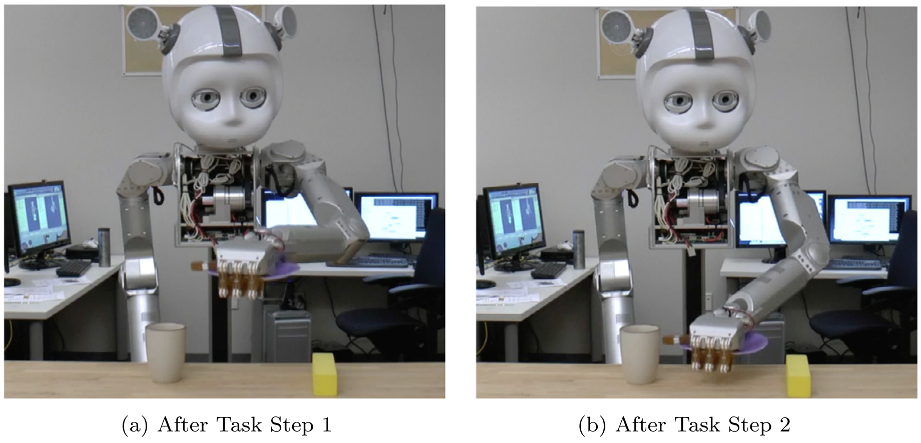


Fig. 5. Steps Comprising the *Table-Setting* Task. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

its performance over three variations of that task. We test performance on two tasks: a *Table-Setting* task and a *Scooping* task.

5.1. Experimental setup

We evaluated our approach on the Curi robot shown in Fig. 5. Curi is equipped with two arms consisting of 7 degrees-of-freedom and an under-actuated gripper. We used only the robot's left arm for demonstrations and task execution. During demonstrations, we used a gravity-compensating controller so that the robot's arm could be easily moved to complete the task. The robot perceived its tabletop workspace using an overhead RGBD camera (not pictured) which provided a top-down view of the table.

For this experiment, we defined each element of the TTA representation as follows:

- Action Models: We demonstrated each task on a single, 7 DOF arm on the robot shown in Fig. 5. Each demonstration was recorded as a single, continuous trajectory which was then segmented manually into several task steps. We trained a task model over each step as a Dynamic Movement Primitive (DMP), which can be re-parameterized for a new task configuration by specifying the new start and end poses for the desired trajectory.
- Parameterization Functions: We defined the parameterization function for each task model as the end-effector's position with respect to the nearest object in the robot's environment. This reflects the constraints guiding the end-effector's start and end position at each step of the task as an offset from an object location. For example, suppose that one segment of a scooping task ends with the robot's end-effector positioned 5 cm above the pasta bowl before continuing with the next task step. The corresponding parameterization function is: $<o_x, o_y, o_z + 5>$, where o is a reference to the relevant object (in this case, the location of the pasta bowl). The robot recorded these object poses (and subsequently, the parameterization functions with respect to those object poses) autonomously using an RGBD camera located above its tabletop workspace. When transferring the TTA representation at an abstraction where the parameterization function must be grounded in the target environment, we manually re-define this 3D offset.
- Feature Selectors: The robot assigns a unique, non-descriptive object ID to the segmented objects in its environment. These object IDs are referenced in the aforementioned parameterization functions. When transferring the TTA representation at an abstraction where the feature selectors must be grounded in the target environment, we manually provide a one-to-one mapping between object IDs in the source and target environments.
- Feature Values: These are the feature values associated with each object label. We define these as the bounding box dimensions and position of each object in the robot's environment. We use the algorithm described in [39] to segment the RGBD pointcloud into a set of bounding boxes surrounding each object above the tabletop surface plan. When transferring the TTA representation at an abstraction where the feature values must be grounded in the target environment, the robot autonomously updates the feature values from its RGBD sensor input.

5.2. Table-setting task

In the first experiment, the robot learned a table-setting task in the environment shown in Fig. 6a. The table-setting task consisted of placing a plate between a cup and utensil (represented by the yellow block), requiring two skill models encoding (i) moving the end effector to a point between the cup and utensil (Fig. 5a), and (ii) setting the plate down (Fig. 5b).

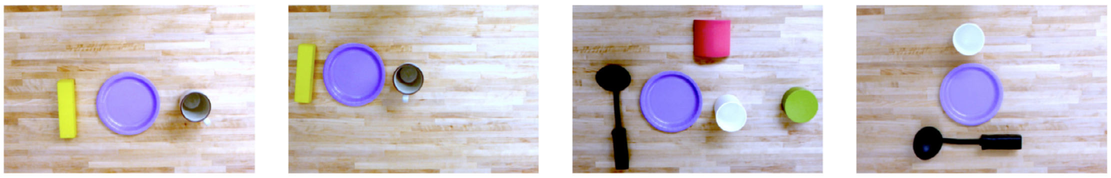

(a) Demonstration State (b) Displaced Objects (c) Replaced Objects (d) Rotated Scene

Fig. 6. Variants of the *Table-Setting* Task Environment.

5.2.1. Training

The training portion of the experiment was run as follows:

1. At the start of the demonstration, the 3D position of each object was recorded via an overhead RGBD sensor, resulting in a 3 x n representation of n object positions.
2. Throughout the demonstration, the robot recorded its 7D end-effector pose (consisting of position $< x, y, x >$ and orientation $< qw, qx, qy, qz >$) relative to the robot's base pose.
3. We manually segmented the demonstration using voice commands, resulting in two separate trajectories: one to position the plate, and another to lower it onto the table. Each trajectory was recorded as a 7 x k representation of the robot's 7D end-effector pose at each of k keyframes recorded throughout the trajectory.
4. At the end of each task segment, the robot recorded the 3D position transform between the end-effector and the object closest to it.
5. Following the demonstration, we trained a DMP over each of the two trajectories.

5.2.2. Testing

We evaluated task performance on three transfer categories:

1. *Displaced-Object environments*: Contain the same objects as in the original demonstration, but displaced as shown in Fig. 6b.
2. *Replaced-Object environments*: Contain cup and utensil objects that are different than those used in the original demonstration. Additional "distractor" objects are also provided that are irrelevant to completing the skill. An example is shown in Fig. 6c.
3. *Rotated-Scene environments*: Contain the cup and utensil objects as in the replaced-object environment, but with the cup and utensil jointly rotated 45-90 degrees away from the robot. An example of this is shown in Fig. 6d. This has the effect of requiring that the robot fulfill the object constraint of placing the plate between the two other objects, rather than simply placing the plate to the left of the cup as in the previous target environments.

These categories of target environments correspond to the feature sets listed in Fig. 4. We represented the task model at *three* levels of abstraction, and evaluated each abstraction on *ten* target environment variations in each of the *three* environment categories, resulting in a total of *90* transfer evaluations for the table-setting task. A "success" was noted each time the plate was placed between the cup and utensil without the plate touching either object.

5.3. Scooping task

The second experiment revisited the scooping task environment depicted in Fig. 8a. The scoop task consisted of four skills: moving the scoop from the initial position at the robot's side to the pasta bowl (Fig. 7a), scooping the pasta (Fig. 7b), moving the scoop to the target bowl (Fig. 7c), and then pouring the scoop over the target bowl (Fig. 7d).

5.3.1. Training

We performed the training portion of this task similar to that of the table-setting task, but with the robot grasping the yellow scoop prior to starting the demonstration. Since the scooping task is more complex than the first task, we recorded three demonstrations, keeping the demonstration that yielded the most stable performance when re-tested in the source demonstration environment.

5.3.2. Testing

We evaluated the robot's performance in three transfer categories:

1. *Displaced-Object environments*: Contain the same objects as in the original demonstration, but displaced as shown in Fig. 8b.

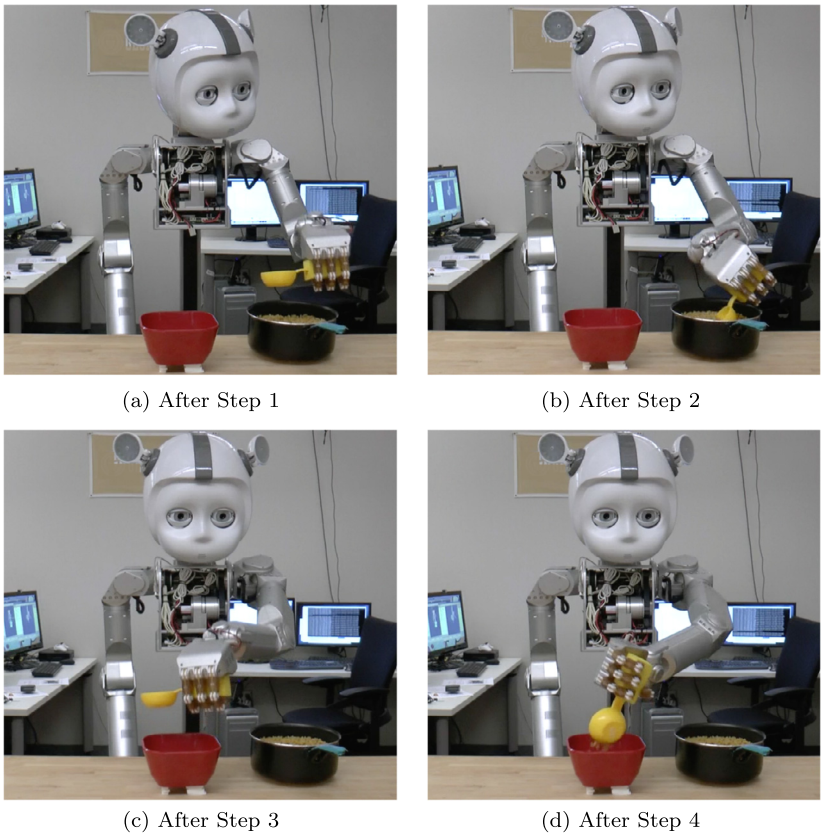

(a) After Step 1 (b) After Step 2 (c) After Step 3 (d) After Step 4

Fig. 7. Steps Comprising the *Scooping* Task.

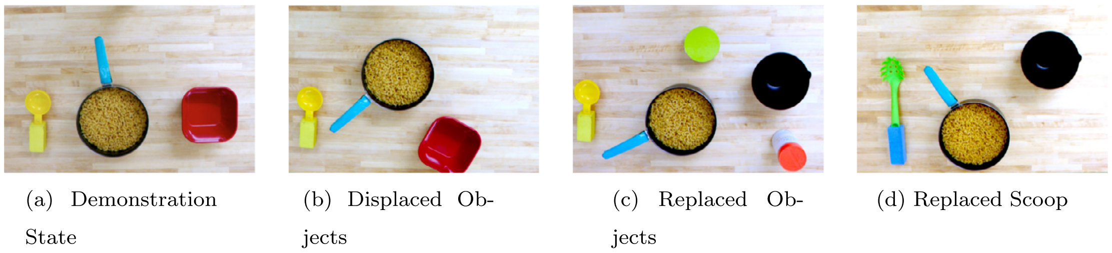

(a) Demonstration State (b) Displaced Objects (c) Replaced Objects (d) Replaced Scoop

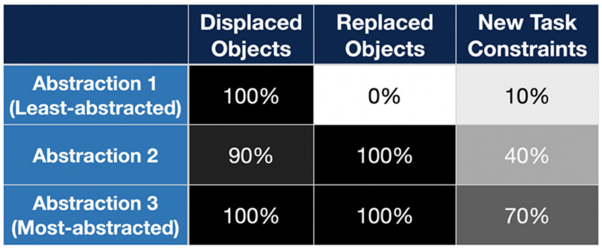
Fig. 8. Variants of the *Scooping* Task Environment.

2. *Replaced-Object environments*: Contain a different target bowl and a set of additional, "distractor" objects that are irrelevant to completing the task, as in Fig. 8c.
3. *Replaced-Scoop environments*: Contain the same target bowl as in the replaced-object environment, but also contain one of two scoops with longer handles than the one used in the original demonstration, as in Fig. 8d.

The *three* task abstractions were each applied to transfer the task *ten* times per each of the *three* environment categories, resulting in a total of *90* transfer evaluations for the scooping task. A "success" was noted each time any amount of pasta was moved to the target bowl without the target bowl being tipped over.

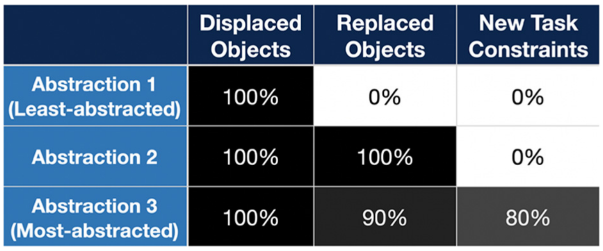
5.4. Results: applying the tiered task abstraction to task variations

Fig. 4 summarizes the three abstraction levels evaluated in this experiment. For both the table-setting and scooping tasks, we hypothesized that (i) Abstraction 1 could only consistently address the displaced-objects environment, (ii) Abstraction 2 could consistently address the displaced and replaced objects environments, and (iii) Abstraction 3 could consistently address all three environments.



	Displaced Objects	Replaced Objects	New Task Constraints
Abstraction 1 (Least-abstracted)	100%	0%	10%
Abstraction 2	90%	100%	40%
Abstraction 3 (Most-abstracted)	100%	100%	70%

Fig. 9. Success Rates for Each Abstraction Level Applied to the Table-Setting Task.



	Displaced Objects	Replaced Objects	New Task Constraints
Abstraction 1 (Least-abstracted)	100%	0%	0%
Abstraction 2	100%	100%	0%
Abstraction 3 (Most-abstracted)	100%	90%	80%

Fig. 10. Success Rates for Each Abstraction Level Applied to the Scooping Task.

5.5. *Table-setting task*

Fig. 9 provides the success rate of each transfer method when applied to 10 varying instances of each category of table-setting environments. As expected, Abstraction 1 succeeded consistently on *only* the displaced-objects environment. Abstraction 2 resulted in consistent performance in the first two environments, and additionally, succeeded in a few of the rotated-scene scenarios. Transfer at this abstraction level succeeded in the few occasions when the robot was able to place the plate to the left of the cup without the plate touching either the cup or utensil. This demonstrates that while this abstraction level may be used to address some of the rotated-scene environments, it cannot do so consistently.

In the three scenarios in which Abstraction 3 did not succeed in addressing a rotated-scene environment, the front of the robot's hand had hit the cup, leaving the plate close to the intended location but not quite meeting the threshold for successful task completion. We anticipate that this abstraction could be used more successfully if the parameterization function used to ground this abstraction had incorporated information about the size of the cup and the robot's hand to avoid hitting other objects.

5.6. *Scooping task*

Fig. 10 provides the success rate of each abstraction level when applied to target environments in three categories of scooping task problems. As in the previous results, the displaced-objects environments could be addressed consistently using any of the three abstraction levels, and the replaced-objects environment could only be addressed consistently by Abstractions 1 and 2. Finally, these results also indicate that Abstraction 3 succeeded consistently across all three classes of transfer problems.

6. Grounding: trade-off between generality and data-efficiency

These results suggest that Abstraction 3 provided the most consistently successful results across the full range of transfer problems we tested, with Abstractions 2 and 1 each addressing fewer transfer problems, respectively. However, the more that the task representation is abstracted, the more data is required to ground that abstraction in the target environment. Abstraction 3 requires both an object mapping and parameterization function in order to ground this abstraction in the target environment. While this grounded data was provided manually in this experiment, we intend for the robot to eventually learn this data either autonomously or from more indirect assistance (such as a human teacher providing additional task demonstrations or answering the robot's questions). These results indicate a **trade-off between the generality of a task representation, and the amount of additional information required to ground that task representation in the target environment**. Furthermore, these results support our hypothesis that there is a **correlation between (i) the level of similarity between the source and target environments and (ii) the level of abstraction that should be used to address a transfer problem**.

We also note that abstraction occurs within the DMP action model itself. DMPs are intended to control the robot's end-effector position, which is itself an abstraction of the robot's motion in joint-space. Furthermore, DMPs model an end-effector trajectory as a point-attractor system that is perturbed by the centering and weighting of several basis functions that are temporally activated throughout the trajectory. As a result, the point-attractor system enables the trajectory to be guided by a start and end position that may be modified, while also maintaining the general shape of the trajectory. This enables an abstraction of the goal parameterization that is separate from the dynamics parameters, making it ideal for use in the TTA representation. While another action model may be used, it may need to be accompanied by an additional parameterization function in order to enable this separation of goal parameters from dynamics parameters.

6.1. Grounding task abstractions via interaction

Fig. 4 summarizes the representation elements which must be retained or grounded for each category of transfer problems. Our experiment demonstrates the effectiveness of each abstraction level on a range of task variations, with each abstraction being grounded manually. For a robot that operates in human environments, we aim for the robot to be able to ground its own task abstractions using continued interaction with a human teacher during task transfer. This would enable the robot to leverage the human teacher's knowledge of the task domain in order to perform transfer. This relationship between (i) environment similarity and (ii) assistance from the human teacher introduces a second dimension to the aforementioned similarity spectrum; **as the source and target environments become more dissimilar, the robot's level of transfer autonomy decreases and its dependence on interaction with the human teacher increases**.

As discussed in the previous section, the first two categories of transfer problems (e.g. identical and displaced-objects environments) could be addressed by the robot with full autonomy. In order to address more difficult transfer problems, the robot must ground both the (i) parameterization functions and (ii) skill models in the target environment. These are the two elements of the TTA representation which contain the most high-level information about the task: the constraints between the robot's end-effector and objects in the environment, and the action model which preserves the trajectory shape of the demonstrated action, respectively. These represent high-level information about the task, and require knowledge of the task goal to define. As a result, we do not expect that this data can be grounded by the robot with complete autonomy, but rather, may be obtained using input from a human teacher.

The aim of interactive grounding is to produce a solution that (i) is partially autonomous (the robot interacts with a human teacher and may receive additional instruction, but does not require a full re-demonstration of the task), (ii) enables collaboration with the human teacher so that the robot may infer information about the task in the target environment, (iii) results in parameterization functions and/or action models that can ground an abstracted task representation, and (iv) grounds the TTA representation such that a trajectory can be executed in the target environment.

In prior work, we have defined two methods for interactive grounding, each targeting a different category of transfer problems. In [40], we presented Mapping by Demonstration in which a robot utilizes a targeted method of interaction with a human teacher (indicating the next object the robot should use) in order to infer the object feature that dictates the object mapping within the context of a particular task. This enables the robot to learn to predict the mapping using limited assistance with the first part of the task, and then transfer the remainder of the task autonomously. The resulting object mapping enables the robot to ground Abstraction 2 and thus address transfer problems with replaced objects.

We have also presented a method for grounding Abstraction 3 for transfer problems in which object replacements alter the manipulation constraints of the task. To address this category of transfer problems, a different interaction mode is needed to ground the relationship between (1) the new object and (2) the trajectory adaptations necessary to use the new object. In [41], we employed *corrections* to record and model constrained points in the robot's motion. Furthermore, we presented a method for modeling the new constraints afforded by the tool within the context of the corrected task, and demonstrated that the learned model can also be reused on other tasks that provide a similar context for that tool (e.g. in the tool surfaces used to execute the task).

By viewing interaction as a means of grounding task abstractions, we can leverage additional, existing methods for interactive robot learning. Recent work on learning from preferences [42], critiques [43], and dialogue [44] demonstrates that a robot can effectively learn from these widely different types of interactions, while also differing in the quantity and specificity of the data that is derived from each type of interaction. As a result, the relationship between (i) the level of abstraction used to represent the task and (ii) the information needed to ground the abstracted representation thus **provides guidance for selecting the interaction modality that is best suited for a particular transfer problem.**

7. Conclusion

In this article, we have contributed the TTA representation which:

1. Addresses transfer not as a single problem, but rather as a series of problems that range in difficulty according to the similarity between the source and target environments.
2. Defines a relationship between (i) the similarity between source and target environments, (ii) the effect of this similarity (or dissimilarity) on the robot's task execution, and (iii) the level of abstraction at which the task model should be represented in order to enable transfer to that target environment.
3. When abstracted, is capable of addressing transfer problems with varying dissimilarity between the source and target environments.

4. Provides experimental results that illustrate the effect of transferring a task model at several levels of abstraction, and the resulting performance over a set of transfer problems ranging in their difficulty.

7.1. Key contributions and insights

This article analyzes task transfer in the data-sparse context of robot learning from demonstration. While abstraction-based approaches to task transfer have been addressed via cognitive systems research, to our knowledge this article is the first to both apply an abstraction-based approach to the problem of robotic task transfer and ground it in action, perception, and interaction. Furthermore, while prior work has shown how interaction can be used to ground specific abstractions of a task representation, the primary contribution of our work is a generalized, TTA representation that can be abstracted and applied to a range of transfer problems.

Through a case study consisting of two tasks, we have evaluated transfer performance using three levels of abstraction, each of which was grounded manually for the target environment. This serves to answer the question of *whether* we can use a single task representation to address a range of transfer problems via abstraction.

Insight #1: The more dissimilar the source and target environments are, the more that the source task representation must be abstracted to be successfully transferred to the target environment.

Insight #2: There is an inverse correlation between (i) the degree to which the task representation is abstracted and (ii) the amount of data that is needed to ground the abstracted representation in the target environment.

Insight #3: As a result of #1 and #2, there is a tradeoff between the generality of a task representation (e.g. the range of transfer problems that it can successfully address) and the data requirements that must be met to ground the abstracted task representation in the target environment.

Insight #4: The modality of the interaction (such as gestures, corrections, or descriptive dialogue) between the robot and human teacher has a direct impact on the information the robot can derive from the teacher's feedback. As a result, the grounding requirements of the abstracted task representation must be taken into consideration when selecting the interaction modality used to obtain that grounded data.

7.2. Open questions

While we have defined the features relevant to classifying the similarity between a source and target environment, it is a different matter for the robot to detect these features autonomously. Furthermore, turning these features into a classifier or quantifiable similarity metric remains an open challenge. Such a classifier would enable the robot to identify *which* of its previously-learned task models are most suitable for addressing a novel task and/or environment. While similar problems have been addressed via case-based retrieval (see Section 2.2), this method has not yet been applied to a robotics domain; particularly, the problem of identifying state-space changes in task transfer problems.

Furthermore, we have assumed that the robot assesses the similarity between source and target environments a priori, and then utilizes the corresponding level of abstraction for the entirety of the task. Rather than utilize a static abstraction for the full task, there is an opportunity for future research to consider a dynamic task abstraction that is updated at each step of the task. This would enable the robot to select the appropriate task representation (and subsequent interaction for grounding) based on the task differences that are relevant to a single step of the task. In doing so, the robot may tailor the frequency and method of interaction it requests so as to (i) model the exact information needed for transfer, and (ii) maximize its overall autonomy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This material is based on work supported by the NSF Graduate Research Fellowship under Grant No. DGE-1148903.

References

[1] J.E. Laird, K. Gluck, J. Anderson, K.D. Forbus, O.C. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, G. Trafton, et al., Interactive task learning, IEEE Intell. Syst. 32 (4) (2017) 6–21.
[2] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: a survey, J. Mach. Learn. Res. 10 (2009) 1633–1685.
[3] C. Martinie, D. Navarre, P. Palanque, C. Fayollas, A generic tool-supported framework for coupling task models and interactive applications, in: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2015, pp. 244–253.
[4] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robot. Auton. Syst. 57 (5) (2009) 469–483.
[5] S. Chernova, A.L. Thomaz, Robot learning from human teachers, Synth. Lect. Artif. Intell. Mach. Learn. 8 (3) (2014) 1–121.
[6] B. Akgun, M. Cakmak, J.W. Yoo, A.L. Thomaz, Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective, in: ACM/IEEE International Conference on Human-Robot Interaction, ACM, 2012, pp. 391–398.

[7] S. Schaal, Dynamic movement primitives-a framework for motor control in humans and humanoid robotics, in: Adaptive Motion of Animals and Machines, Springer, 2006, pp. 261–280.
[8] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2009, pp. 763–768.
[9] S. Chernova, M. Veloso, Confidence-based policy learning from demonstration using Gaussian mixture models, in: International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, ACM, 2007, p. 233.
[10] D. Bruno, S. Calinon, D.G. Caldwell, Learning adaptive movements from demonstration and self-guided exploration, in: Joint IEEE International Conferences on Development and Learning and Epigenetic Robotics, ICDL-Epirob, IEEE, 2014, pp. 101–106.
[11] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373.
[12] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, W. Zaremba, One-shot imitation learning, in: Advances in Neural Information Processing Systems, NeurIPS, 2017, pp. 1087–1098.
[13] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, C. Finn, Universal planning networks, in: International Conference on Machine Learning, ICML, 2018.
[14] J. Fu, S. Levine, P. Abbeel, One-shot learning of manipulation skills with online dynamics adaptation and neural network priors, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2016, pp. 4019–4026.
[15] T.W. Killian, S. Daulton, G. Konidaris, F. Doshi-Velez, Robust and efficient transfer learning with hidden parameter Markov decision processes, in: Advances in Neural Information Processing Systems, NeurIPS, 2017, pp. 6250–6261.
[16] J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, 1993.
[17] B. Falkenhainer, K.D. Forbus, D. Gentner, The structure-mapping engine: algorithm and examples, Artif. Intell. 41 (1) (1989) 1–63.
[18] P. Thagard, K.J. Holyoak, G. Nelson, D. Gochfeld, Analog retrieval by constraint satisfaction, Artif. Intell. 46 (3) (1990) 259–310.
[19] A.K. Goel, Design, analogy, and creativity, IEEE Expert 12 (3) (1997) 62–70.
[20] A.K. Goel, S.R. Bhatta, Use of design patterns in analogy-based design, Adv. Eng. Inform. 18 (2) (2004) 85–94.
[21] H.W. Park, A.M. Howard, Case-based reasoning for planning turn-taking strategy with a therapeutic robot playmate, in: IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob, IEEE, 2010, pp. 40–45.
[22] M.W. Floyd, B. Esfandiari, A case-based reasoning framework for developing agents using learning by observation, in: IEEE International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, 2011, pp. 531–538.
[23] A.-M. Olteţeanu, Z. Falomir, Object replacement and object composition in a creative cognitive system. Towards a computational solver of the alternative uses test, Cogn. Syst. Res. 39 (2016) 15–32.
[24] D.K. Misra, J. Sung, K. Lee, A. Saxena, Tell me Dave: context-sensitive grounding of natural language to manipulation instructions, Int. J. Robot. Res. 35 (1–3) (2016) 281–300, https://doi.org/10.1177/0278364915602060, http://journals.sagepub.com/doi/10.1177/0278364915602060.
[25] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J.M.M. Montiel, A. Perzylo, et al., A world wide web for robots, IEEE Robot. Autom. Mag. 18 (2) (2011) 69–82.
[26] J.R. Kirk, J.E. Laird, Learning hierarchical symbolic representations to support interactive task learning and knowledge transfer, in: IJCAI, 2019, pp. 6095–6102.
[27] M. Tenorth, D. Nyga, M. Beetz, Understanding and executing instructions for everyday manipulation tasks from the world wide web, in: IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2010, pp. 1486–1491.
[28] K. Bullard, B. Akgun, S. Chernova, A.L. Thomaz, Grounding action parameters from demonstration, in: IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN, 2016.
[29] B. Akgun, A. Thomaz, Simultaneously learning actions and goals from demonstration, Auton. Robots 40 (2) (2016) 211–227.
[30] S. Ekvall, D. Kragic, Learning task models from multiple human demonstrations, in: IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN, IEEE, 2006, pp. 358–363.
[31] J. Kirk, A. Mininger, J. Laird, Learning task goals interactively with visual demonstrations, Biol. Inspir. Cognit. Archit. 18 (2016) 1–8.
[32] G. Konidaris, S. Kuindersma, R. Grupen, A. Barto, Robot learning from demonstration by constructing skill trees, Int. J. Robot. Res. 31 (3) (2012) 360–375.
[33] M.E. Taylor, S. Whiteson, P. Stone, Transfer via inter-task mappings in policy search reinforcement learning, in: International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, ACM, 2007, p. 37.
[34] J. Sinapov, A. Stoytchev, Detecting the functional similarities between tools using a hierarchical representation of outcomes, in: IEEE International Conference on Development and Learning, ICDL, IEEE, 2008, pp. 91–96.
[35] S. Brown, C. Sammut, Tool use and learning in robots, in: Encyclopedia of the Sciences of Learning, Springer, 2012, pp. 3327–3330.
[36] T. Fitzgerald, A. Goel, A. Thomaz, Human-robot co-creativity: task transfer on a spectrum of similarity, in: International Conference on Computational Creativity, ICCC, 2017.
[37] S. Calinon, A. Billard, A probabilistic programming by demonstration framework handling constraints in joint space and task space, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 367–372.
[38] H. Hu, L. Chen, B. Gong, F. Sha, Synthesize policies for transfer and adaptation across tasks and environments, in: Advances in Neural Information Processing Systems, NeurIPS, 2018, pp. 1176–1185.
[39] A.J.B. Trevor, S. Gedikli, R.B. Rusu, H.I. Christensen, Efficient organized point cloud segmentation with connected components, in: Semantic Perception Mapping and Exploration, 2013.
[40] T. Fitzgerald, A. Goel, A. Thomaz, Human-guided object mapping for task transfer, ACM Trans. Hum.-Robot Interact. 7 (2) (2018) 17, https://doi.org/10.1145/3277905, http://doi.acm.org/10.1145/3277905.
[41] T. Fitzgerald, E. Short, A. Goel, A. Thomaz, Human-guided trajectory adaptation for tool transfer, in: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1350–1358.
[42] D. Sadigh, A.D. Dragan, S. Sastry, S.A. Seshia, Active preference-based learning of reward functions, in: Robotics: Science and Systems, 2017.
[43] Y. Cui, S. Niekum, Active reward learning from critiques, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 6907–6914.
[44] M. Scheutz, E. Krause, B. Oosterveld, T. Frasca, R. Platt, Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017.