University of Nevada
Reno

Learning to Generalize from Demonstrations

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science

by

Kathryn M. Browne

Dr. Monica Nicolescu, Thesis Advisor

May 2013

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

KATHRYN MAY BROWNE

entitled

Learning To Generalize From Demonstrations

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Monica Nicolescu, Ph. D., Advisor

Mircea Nicolescu, Ph. D., Committee Member

Thomas Quint, Ph. D., Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

May, 2013

Abstract

Learning by demonstration is a natural approach that can be used to transfer knowledge from humans to robots, similar to how people teach each other through examples. To date, numerous approaches have been developed for learning by demonstration, focusing on two main aspects of the learning problem: the teaching of "motor skills" and the transfer of high-level knowledge of "tasks". This thesis focused on the topic of high-level task learning. Currently, approaches that address this problem utilize the assumption that the task to be learned consists of a precise sequencing of steps that need to be executed. The methods, therefore, aim to accurately reproduce an exact copy of the provided demonstration. However, these methods may suffer from overspecializations or misinterpretations of the task to be learned, if there is any variation in the training example. This variation could be due to the fact that demonstrations can be affected by noise or even by significant changes in task structure from one training example to another. This thesis presents an approach to addressing this challenge through generalization, from a small number of demonstrations of the same task. The aim is to extract a task representation that encodes the essential information from all the training examples, in the presence of small or even large variation in the training examples. The proposed solution consists of two main components: a representation that enables the learner to store the generalized representation of the task and the learning algorithm that allows the construction of a generalized task representation. The approach is validated in simulation and on a physical robot working in the real world, showing the ability to generalize to a wide range of scenarios that may typically occur in teaching by demonstration.

Acknowledgments

I would like to thank my thesis advisor Dr. Monica Nicolescu for all of her guidance and support. I would also like to thank Dr. Mircea Nicolescu and Dr. Thomas Quint for taking the time to serve on my thesis committe. A special thanks goes to Brian Hamilton who provided all the computer vision for this thesis. And thanks to all my friends who made me take much needed breaks. A final thanks goes to my parents. I would not be here without all of their support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ability to learn new knowledge is essential for any robot to be successful in real-world applications, where it would be impractical for a robot designer to endow it with all the necessary task capabilities that it would need during its operational lifetime. Therefore it becomes necessary that the robot is able to acquire this information in a human-like teaching approach. The method is also appealing due to the fact that it enables users to endow robots with new capabilities, without the need for the users have computer science or programming background. Successful development of such learning by demonstration mechanisms would open the door for robots to become widely used in daily tasks.

To date, numerous approaches have been developed for learning by demonstration, focusing on either learning of "motor skills" [5] or "high-level tasks" [24]. While significant research has been performed in the area of learning motor behavior primitives from a teacher's demonstration, the topic of learning general "task knowledge" has not been sufficiently addressed. In this field, existing approaches are limited to learning simple tasks and are prone to overspecializations or misinterpretations of the task to be learned. These methods work under the assumption that the only important variation present in the training examples is due to noise in the observations of the teacher. The task to be learned is considered to be a precise sequencing of steps that would need to be executed, and thus the focus is placed on having a very accurate mechanism for observation and interpretation of a teacher's actions. While the development of the sensing module of a learning system is an important and necessary

component, another major challenge comes from the large variation that is typically present in the training examples for real-world, practical applications. In many cases, the same exact task can be demonstrated in a wide variety of ways. For example, "Setting-the-table" would consist of "clear-table", followed by "set-table-cloth", then "set-forks", "set-knives", "set-plates", "set-glasses", followed by "bring-food". This task could be performed and demonstrated in many different ways, in that the forks, knives, plates and glasses can be set in any order, thus generating wide changes in the training examples. Furthermore, other steps could also occur sometimes while setting the table, such as "wash-hands" or "answer-phone", which do not happen in all cases. However, in all demonstrations, the first two steps would happen in the same order ("clear-table", followed by "set-table-cloth") and the "bring-food" will always happen last, after everything needed is on the table. Presented with multiple demonstration of such a task, the main challenge for the learner is to extract all the necessary information pertaining to the task, eliminate all the observations that are irrelevant and generalize the correct task representation in the case when multiple, possibly different demonstrations are given.

This thesis proposes a solution for the above generalization problem: a "representation" that enables the learner to store the generalized representation of the learned task in the form of a precondition graph and a "learning algorithm" that enables generalization of correct and complete task knowledge from a small number of demonstrations of the same task, under the assumption of possibly very different demonstrations provided to the learner. The approach assumes that the learner robot is equipped with a basic set of capabilities or skills; the robot is aware of the goals each of these skills accomplish and also under which conditions these skills can or should be executed. The robot will learn how to use his existing skill repertoire to perform similar tasks or to achieve similar goals as those presented by a human teacher, from a small number of examples. The validation of the presented method is composed of two parts: a learning part and an execution part. During the learning part, the learner is presented with a set of training examples with which it makes a

generalized representation of the task. These generalized representation is then used during execution to generate a sequence of steps that achieves the task that has been demonstrated. A significant advantage of the proposed method is the generalized representation is always consistent with the training samples. That is, the system will never produce an execution sequence that does not obey all the rules represented in the training data. In addition, our approach allows the robot to execute the learned tasks in collaboration with a human user: the user can help the robot by performing some parts of the task and the robot takes into account the user's actions in order to finish up the task. Furthermore, the generalized representation can be used to prevent a learning robot from performing a skill that is mis-representative of the current task. Lastly this work has been published in [4].

The remainder of this thesis is structured as follows: **Chapter 2** summarizes related work in the field of learning by demonstration. **Chapter 3** presents the generalization problems this thesis is attempting to address and it describes the representation and the generalization algorithm. **Chapter 4** presents the experimental results, **Chapter 5** discusses these results and future work, and finally **Chapter 6** concludes the thesis.

Chapter 2

Related Work

2.1 Task Learning

Existing approaches for task learning from demonstration have focused mostly on learning manipulation or assembly tasks using humanoid robots [12]. The representations built in these cases may be constructed in terms of high-level operators constituting capabilities of the robot [14] or as sequences of events that the robots aim to reproduce at the time of the execution. Via-point representations of robot trajectories have also been used in humanoid learning of playing a tennis forehand [19], or the game of kendama [20]. In these approaches, the use of a bi-directional theory for robot control employs a flow of information between the motor command and the trajectory planning levels, which helps to coordinate the high-level decisions with the reactive motor responses.

[28] use Hidden Markov Models (HMM) to learn an egg-flipping task, built from the robot's "grasp, carry, press" and "slide" skills. The demonstrations, provided through teleoperation, allow the robot to learn the task sequence from a small number of demonstrations of the task. Another example that uses HMMs demonstrates learning of a "peg-in-the-hole skill" [9], in which the states of the model are represented by the possible configurations of the assembly and the transitions correspond to moving the peg with various velocities in up to eight possible directions.

In the mobile robot domain, [10] present a mechanism for learning a slalom task. The robot learns by creating a mapping between features in its visual field

and wheel velocities. However, this solution is highly dependent on the particulars of the environment in which the task is being demonstrated. Since it relies on features detected on walls or doors, the robot would fail to reproduce the task in a different environment, even if the positions of the slalom poles and their relations to each other would be identical to those that are obtained during training.

Task learning from demonstration has also been addressed for authoring Intelligent Tutoring Systems (ITS) [21]. Since it is difficult to design autonomous tutors by manually encoding the knowledge of experts in a particular domain, systems for authoring ITS focus on learning this complex procedural knowledge and plans from demonstrations provided by the experts. This is a very challenging problem, as the tutor needs a lot of information in addition to just being able to demonstrate a procedure to students (e.g., monitoring the students, recognizing valid sequences of actions and answering student's questions). The approach presented by [2] uses both training examples and autonomous experimentation to learn hierarchical task knowledge, while in the same time acquiring general-purpose operators that contain reusable domain knowledge. The advantage of using experimentation is that it provides the learning agent with more data for learning, thus helping refine operator preconditions, and reducing the instructor's effort in providing the demonstrations.

In the area of high-level task learning, [22] propose an approach to learning "skills" represented as superposition/fusion of innate robot primitives and of "tasks" consisting of sequencing of the learned skills. More complex tasks can be encoded in hierarchical representations, which consist of abstractions of lower-level tasks.

2.2 Generalization and Learning from Multiple Demonstrations

The majority of approaches that rely on learning from multiple demonstrations are focused on performing function approximations for trajectory or movement primitive learning. Only a few methods to date have addressed this issue at the task level.

[26] use a Hidden Markov Model (HMM) representation of robot arm trajectories

in order to group them into independent clusters representing separate movement primitives. HMMs have also been applied for recognition of hand-written digits [33], among other tasks.

A connectionist approach for learning "wall-following" and "corridor-centering" skills for a Nomad Super Scout robot is presented in [16]. However, since the method relies on continuous consistency in the training sensory data, characteristic of goal-maintenance type of skills, the approach is not able to learn goal-achievement capabilities. [8] designed a neural network architecture for learning to imitate sequences of movements, inspired from the memory processes in the brain. The method allows a KOALA mobile robot to learn to perform motion zigzag, square and star trajectories by following another robot teacher, but is dependent on very accurate teacher following techniques.

Another representative connectionist example is ALVINN [27]. This system trained an artificial neural network to map camera images to steering directions taken by a human driver, in order to design an autonomous land vehicle. ALVINN learned a significant level of skill in driving on various roads and under different conditions after only 10 minutes of training.

In task-level learning, [28] use HMMs to learn a task sequence from multiple demonstrations. [25] present an approach to learning grasping manipulations, based on a small number of demonstrations. In their method, tasks are represented as sequences of interactions between a robot hand and source and target objects. Multiple demonstrations of the same task (i.e., sequences of hand-object interactions) are generalized using a dynamic programming approach for multiple sequence alignment [7]. This approach processes in batch all the examples given, and returns the sequence of task interactions that are present in all the examples. The common sequence of "essential interactions" constitutes the generalized representation for the task. This method is similar to the generalization approach proposed in this thesis, in that it looks for the demonstrated task steps that have been present across all demonstrations. However, their approach relies on the fact that the steps essential for the task

are always detected (i.e., they appear in all task models), which may not always be true due to the limited sensing capabilities of the robots and to the partial observability of the environment. This prevents the learning of tasks containing alternative paths of execution, which do not contain the same steps (e.g., "do-A" OR "do-B", as neither of these steps would appear in all training samples.

In [6], Chernova and Veloso propose an approach for learning by demonstration that has elements of generalization similar to those presented in this thesis. Using a multi-robot setup and a ball sorting task, the learning strategy enables the robots to extract individual policies that generalize on how to sort balls of different colors: e.g., "red balls go into the left bin", "yellow balls go into the right bin". The main contribution of their work is that multiple robots can simultaneously learn individual policies, an approach that is novel in the learning by demonstration domain. However, the generalization process is enabled by user defined features, which are task dependent and require knowledge about the task to be learned. The goal of the work presented in this thesis is to enable robots to generalize the main features of the task without any user-defined, task-specific knowledge being provided.

The approach presented in this thesis is close to that presented in [24], which allows for generalizations from multiple training examples and enables refining of the learned task using feedback provided by the teacher. However, the approach assumes that the task to be learned consists of a precise sequence of steps to be executed; the only variations assumed to be possible among the training examples are the demonstration of irrelevant steps (false positives), omission of steps that are relevant (false negatives), and the presence of alternative paths of execution for parts of the task. The problem of generalization has also been addressed in [17], demonstrating how tutelage and social interactions during training enable a robot to generalize knowledge about a task to different task configurations. The work presented in this thesis addresses the generalization problem from a wider perspective, allowing for learning of higher-level conceptual aspects of the task, as described in the introduction.

2.3 Segmentation and Identification of Demonstration Traces

A significant challenge in designing robot systems that learn from demonstration is the interpretation of observations gathered from the instruction, as the robot has to process the continuous stream of data coming from its sensors, then translate it into appropriate skills or tasks. In most cases, this consists of segmenting the data stream into meaningful units, and then mapping them to a set of existing behavior primitives [29]. There is a large spectrum of approaches to the problem of segmentation, including Principal Component Analysis (PCA) [32], gesture interpretation [31], Learning Vector Quantization (LVQ) [28], motion contact [30], [11] and geometric interpretation of the demonstration [15].

The approach presented in this thesis uses inspiration from psychology [18], which indicates that for imitation learning goal states plays a significant role in reproducing the same behavior. To learn the correct task knowledge, the proposed solution is based on the hypothesis that during a demonstration of a task, the environment changes in ways that are consistent with achieving all the intermediate goals that lead to the final desired outcome. Thus, it is assumed that the learner is equipped with a mechanism for observing the goals of all the intermediate steps in the task (as they are demonstrated by the teacher), using methods similar to [23]. A demonstrated task will then consist of a sequence of behaviors that achieve those goals, corresponding to the various steps in the task. The learner then uses multiple demonstrations of the same task to build a generalized representation of the underlying task.

2.4 Use of Graph Representation

Several recent approaches to learning by demonstration also use a graph/tree representation. Feature decision trees have been used to find the next behavior for a soccer player (Avrahami-Zilberbrand & Kaminka 2005 [1]), topological task graphs using longest common sequences has been used in path planning (Abbas & MacDon-

ald 2011 [3]), and skill trees have been successfully applied to the pinball domain (Konidaris, Kuindersma, Grupen, & Barto [13]). All of these approaches differ from the generalization graphs proposed in this thesis. The feature tree in [1] represents the entire learned task and each node in the tree represents a behavior in the task. Thus, the feature tree is a decision tree and the robot chooses the next behavior based on observable environmental conditions. In contrast, in our proposed approach there are multiple graphs, with one graph corresponding to each behavior and holding the preconditionds of that behavior. In [3], the topologicial task graph is also a single graph, which contains all the possible ways to execute a task. Different paths along the graph represent different modalities of performing the task. In [13] each skill tree respresents a different modality that the robot could use to physically perform a behavior. The graphs in our approach encode information about behaviors' applicability conditions during a task.

Chapter 3

Task Learning from Generalizations

3.1 Classes of Generalization Problems

Approaches to learning by demonstration or from observation are significantly influenced by performance of the demonstrator. Furthermore, the same task could be performed and its goals achieved in multiple ways. Depending on the situation or on the teacher, a demonstration of the same task can vary greatly, thus posing significant difficulties for the learner system. The challenge for the learner is to extract all the relevant information pertaining to the task and eliminate all observations that are irrelevant. Furthermore, depending on the situation, the learner may need to learn to achieve a final desired goal (no matter what behaviors it performs to reach it, or in what order), or the learner may need to follow the exact sequence of actions that was demonstrated, or the learn may have to take other aspects of the task into account in order to learn the task correctly. This information is not directly available to the learner - to extract it, the learner needs to have a mechanism for generalizing it from observations. During learning from demonstrations there are various aspects of the task that could and should be refined through generalizations. The goal of this thesis is to address the following generalization problems:

Generalization 1. Learning to generalize among multiple types of ordering constraints between the steps of the tasks. Under this category there are several types

of constraints that may occur, which are detailed below:

1.1 Tasks for which the ordering of the demonstration steps is important. Under this case, the exact demonstrated sequence of steps needs to be reproduced.

1.2 Tasks for which the ordering of the task's steps is not important, but where the achieving the final goal is the main focus of the demonstration. Under this case, what is important is that the goals of all the steps included in the task need to be achieved, irrespective of their order. In such a scenario, multiple ways of achieving the final goal could be possible.

1.3 Tasks which have combinations of ordering constraints with parts in which the ordering is not important (situations 1.1 and 1.2 above are special cases of this category). Under this category, three representative cases have been identified.

1.3.1 A step (or set of steps) have to be executed before another step (or set of steps), but has no ordering constraint with the other steps in the task. For example, behaviors `A` and `B` always have to happen before `C` and `D`, irrespective of the ordering of the other steps.

1.3.2 A sequence of steps has to be executed in strict consecutive order, but has no ordering constraint with the other steps in the task. For example, behaviors `A`, `B` and `C` have to be performed in this strict order, irrespective of when and in what order the other task steps are performed.

1.3.3 A step has to be executed at the same position (in the total ordering of steps). For example, behavior `A` always needs to be performed fifth, irrespective of the ordering of the other behaviors.

Generalization 2. Learning to distinguish which steps in the demonstration are relevant, and which steps do not bring any contribution to the task. This will be done by monitoring the steps that do not appear in every demonstration.

Generalization 3. Learning to distinguish the number of times a step needs to be repeated since this number may change in each demonstration. This will be done by monitoring the postconditions of the repeated step after all the repetitions.

The system proposed in this thesis automatically extracts the information needed for the above generalization problems, without any task knowledge or user provided information. That is, the system can detect if the ordering of steps is relevant or not, which steps are a part of an ordering constraint or of a strict sequence, or detect the fixed position on which a behavior needs to be executed. This extends the capabilities of current learning by demonstration strategies, which are focused mainly on learning a precise representation of a task, assuming only slight variation in the training data (due mainly to perceptual limitations and noise).

3.2 Collaborative Behavior during Task Execution

Another goal of the proposed approach is to handle situations in which a teacher or another user helps out the robot that is currently carrying out a learned task. In this case the learner will need to detect that some of the behaviors may have been done while the learner was performing a different behavior in the task. The learner also needs to make sure the behavior that the other person performed met the postconditions of the behavior and if not, the learner will need to complete that behavior.

3.3 Representation of Generalized Tasks

In this work a task is represented as a sequence of symbols, where each symbol corresponds to a behavior that a robot can perform. A sequence of such symbols, provided from a teacher's demonstration, constitutes a training sample. For the purpose of presenting the generalization strategy proposed here, the system uses training samples generated by hand for the simulation experiments. To obtain a training sample in our physical robot domain, a Kinect camera tracked the teacher's hand and the ob-

jects being manipulated. A behavior is achieved once the teacher's hand came into contact with a tracked object for a certain amount. As the demonstration unfolds the sequence of demonstrated behaviors constitutes a representation of the task to be learned.

In this context, a learner is given a set of sequences, potentially different, that all achieve the same target task. These sequences can be used to build a generalized representation of the task. The generalization strategy is based on the preconditions of behaviors shown in the demonstration, in particular on the set of behaviors that have occurred prior to their execution. For instance, if behaviors `a` and `b` come before `d` in all training samples but `c` can appear before or after `d`, we can generalize that behaviors `a` and `b` are preconditions for behavior `d` as they may achieve goals that are necessary for `d`'s execution. The goal is to build, for each behavior present in the demonstrated task, a generalized representation of behavior preconditions that encapsulates the information contained in all training samples. This representation could afterwards be used by the learner to perform the same observed task, while capturing the variability present in the demonstrations.

The representation chosen to encode the generalized preconditions is a directed graph that is similar to a tree structure with the added feature that leaf nodes in the graph can contain a edge pointing back to the root of the graph. The root of the graph is a behavior for which we build the generalized preconditions and the branches of the graph are the behaviors that came before the current behavior in the training examples. Thus, each behavior in the training example has its own graph.

The graphs are created by adding behaviors that came before the current behavior to the graph. The algorithm starts by taking the first behavior in the first training sample and creating a precondition graph for it. A node named `NULL` is added to the graph as a child, because no behavior has been executed before it, and therefore the behavior has no preconditions. This `NULL` node is marked as a "final state"; being a "final state" signifies that the behavior represented by the root node can be performed after the behavior marked as a "final state" in the current sequence. If

multiple demonstrations are provided, the number of times a final node appears before the current behavior is recorded and used at the execution stage, in order to decide which behavior to perform in a sequence when multiple behaviors are applicable and can be chosen at the same time. Next, the algorithm creates a precondition graph for the second behavior. In this graph, the first behavior is added to it as a child node. This child node is also marked as a final state and the number of times it appeared as a final state is (initially) set to one. The algorithm continues to go through each sequence in the example, creating precondition graphs for all the behaviors, and adding branches as needed.

To demonstrate this method, consider an example in which the following training samples have been given for a particular task (where `a`, `b`, `c` and `d` are behaviors that the robot can execute):

- Demonstration 1: `a b c d`
- Demonstration 2: `c a d b`
- Demonstration 3: `c d b a`

The algorithm begins by creating a new precondition graph, with `a` becoming the root node and a node with the name `NULL` added as a child. This child node is then marked as final state. In addition, the number of times `NULL` has occurred as a final state for `a` is set to one (Figure 3.1, first graph). Next, the algorithm creates the precondition graph for behavior `b`; the graph starts with `b` as the root node and `a` as a final state child node, because `a` was executed before it in the first training sample (Figure 3.1, second graph). A new graph is then created for behavior `c` and the branch with nodes `a` and `b` is added. Finally the graph for `d` is created and `a`, `b`, and `c` are added as a branch. The graphs created are shown in Figure 3.1. The numbers appended to the behaviors in the graphs have the following meaning: the first number appended signifies the level that it is located in the graph and the second number, which is only appended to the final state nodes, represents the number of

times the node was a final state in the training samples (e.g., `c3_1` means that the node is at level 3 and was a final state for `d` only once).

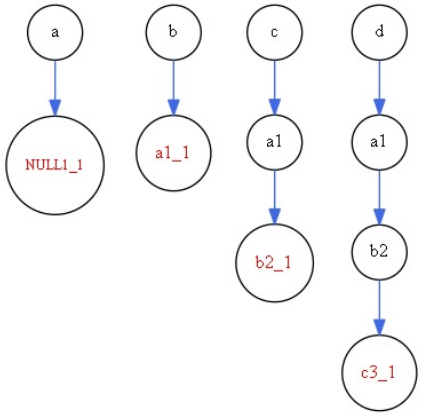

Figure 3.1: Graphs created after the first sequence of behaviors.

After the second demonstration sequence is provided, all the precondition graphs are refined as follows. Behavior `c` is the first in the training sample, so a null child node is added to its graph, indicating that it can also be the first one executed in the task. Then, behavior `c` is added as a child node in behavior `a`'s graph, a branch containing behaviors `c` and `a` is added to `d`'s graph, and finally `c`, `a`, and `d` become a new branch in `b`'s graph. Figure 3.2 shows the graphs after adding the second training sample.

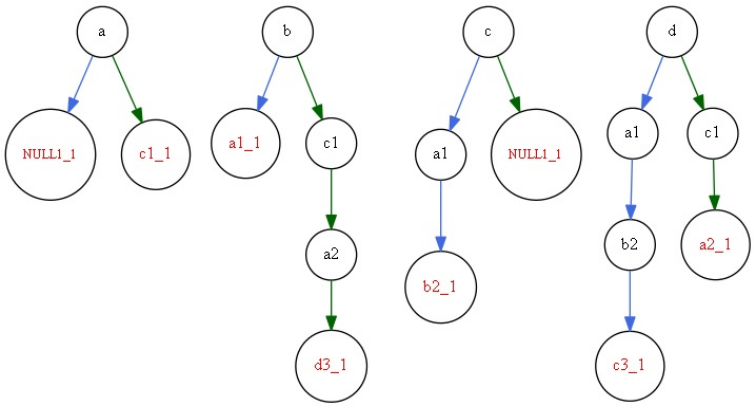

Figure 3.2: Behavior graphs after second sequence.

The third training sample is incorporated using the same strategy as above. The resulting generalized precondition graphs can be seen in Figure 3.3.

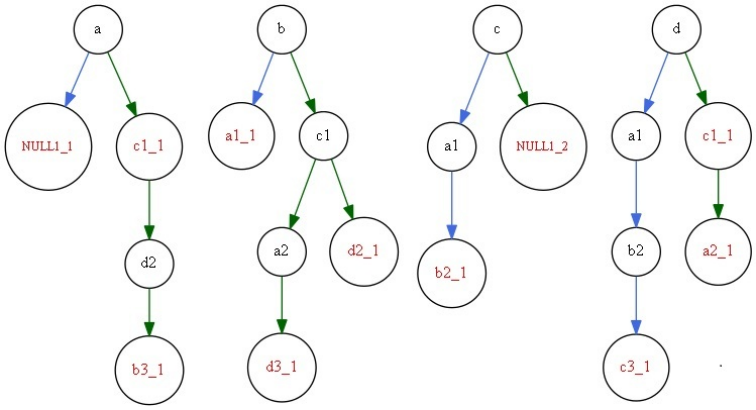

Figure 3.3: Behavior graphs after third and last sequence.

In the general case, a behavior may occur in a training sample more than one time. In such a case, the precondition graphs are built as described before, but with children nodes in the generalized precondition graph pointing back to the parent node. Consider the sequence:

- `a b c f d e f g`

When a graph for behavior `f` is created, a branch containing the behaviors `a`, `b`, and `c` will be added to `f`'s graph and the `c` node will have an edge pointing to `f`'s node. A branch containing `d` and `e` is also added to the graph. `f`'s graph can be seen in Figure 3.4

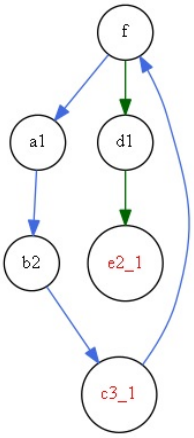

Figure 3.4: Behavior graph whose behavior appears one or more times.

Furthermore, behaviors that have the same behavior for every final node in the

generalized precondition tree can be combined into a single behavior, with a unique precondition tree because the preconditions for the second behavior in the combination are the same as the first behavior. Since the preconditions are the same, combining the graphs simply means appending the second behavior to the first and deleting its own graph. An abstraction of such multiple behaviors is called a "generalized behavior".

To illustrate a generalized behavior graph, consider the following example in which `a` always comes right before `b` in the training samples:

- Demonstration 1: `a b c d e`
- Demonstration 2: `c a b e d`
- Demonstration 3: `e c d a b`

Figure 3.5(left) shows the graph for behavior `a` and Figure 3.5(center) shows the graph for behavior `b` before the behaviors are combined. Figure 3.5(right) shows the generalized behavior that results from combining the behavior graphs `a` and `b`.

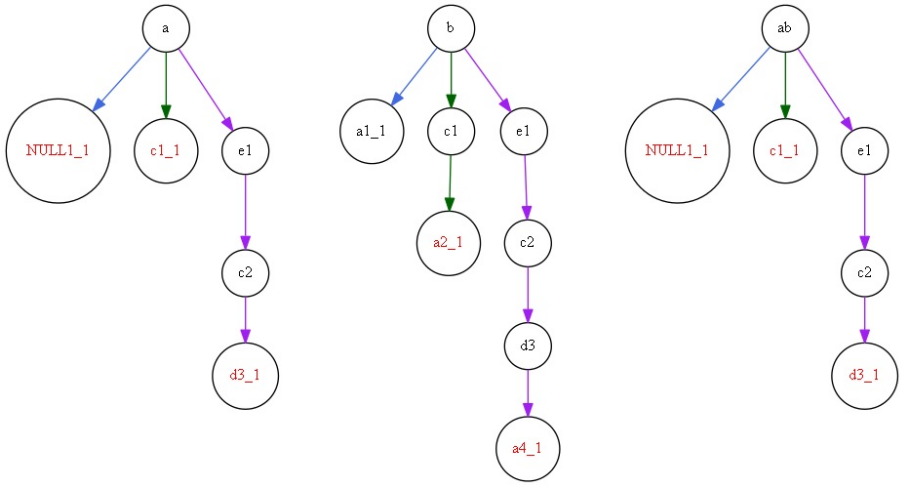

Figure 3.5: Precondition graphs for a(left), and b(center), and generalized behavior ab(right)

Additionally, the graphs keep track of the number of training samples each behavior appears in. This number can be used to calculate the percentage of times a

behavior appeared in the training example, which in turn is used to find behaviors that did not appear in every sample. Depending on this percentage, a choice can be made to keep the behavior's graph or to completely remove it. In this work, a 50% threshold is used for removing a behavior graph, but this percent can easily be changed to a value relevant for different application domains..

In any real-world applications, is sometimes necessary to repeatedly execute a behavior until some condition is met, which can vary from one training sample to another. For example, consider the task of hand washing a dish. Since there is never a consistent amount of dirt on each dish, the amount of times a dish is scrubbed can vary from dish to dish. Thus, it is sometimes necessary to repeatedly execute a behavior until some condition is met, with the number of repetitions of the behavior possibly varying from training sample to training sample. To account for this situation, the generalized precondition graphs also record if a behavior appeared more than one time in a training sample. In our representation, these behaviors are aggregated into "it special behaviors" which are just behaviors that the learner repeats in the task as efficiently as possible until the post conditions are met. For example, if the task required to add two cups of flour, the learner would not add 1/4 of a cup of flour eight times; instead the learner would add one cup of flour twice. At the beginning of the learning period, the learner is provided a lists of behaviors that can be performed and the possible parameters for these behaviors. For instance, a behavior such as add sugar can be performed by adding a half cup, a cup, or two cups.

Using the above representation of demonstrations in the form of precondition graphs leads to more compact encoding of multiple demonstrations and abstracting of key features of the varying examples. These representations have the key feature that they can be used by the robot to generate execution sequences that are always consistent with the training samples. That is, the system will never produce an execution sequence that does not obey all the rules represented in the training data. The following section describes how the precondition graphs can be used toward this purpose.

3.4 Usage of Generalized Precondition Graphs

The generalized representations built through the method presented above can be used by the learner to reproduce the task that has been demonstrated. Given that multiple demonstrations have been provided, the learner needs to find an execution sequence that achieves the same goal and is consistent with all the training examples. To do this, the system needs to find, at any time, a behavior that is applicable (i.e., can be executed), based on the current execution sequence up to that point. To find the applicable behaviors, the learner uses the precondition graphs built from demonstrations as follows: for each precondition graph of all the behaviors, if the current execution sequence up to that point matches a complete branch in the graph (from the first level under the root, down to a final state), then the behavior at the root of the tree can be executed at that time. If two or more behaviors fit the criteria, the behavior that has been visited the most (based on the node counts) is picked, or a behavior is picked at random in case of equality. To demonstrate this, consider the previous example and graphs in Figure 3.3. The sequence below indicates how the precondition graphs are used to decide what behavior to execute at each step, taking into account the history of execution:

t_0 Current execution sequence: `null`

Behaviors that qualify: `a` and `c` Choose: `c` (occurred more often than `a` as first in the task)

t_1 Current execution sequence: `c`

Behaviors that qualify: `a` and `d` (equally likely) Randomly Choose: `d`

t_2 Current execution sequence: `c d`

Behaviors that qualify: `b` Choose: `b`

t_3 Current execution sequence: `c d b`

Behaviors that qualify: `a` Choose: `a`

t_3 Current execution sequence: `c d b a`

Behaviors that qualify: Choose: nothing - task done

Chapter 4

Experimental Results

The system was tested in the context of a task for making the dough for chocolate chip cookies. A training example was a set of sequences that were made up of strings representative for the behaviors needed for this task. The validation of the system consisted of analyzing the precondition graphs and evaluating if the generalized representations produced an execution trace consistent with all the training examples. In most cases, the system produced execution sequences that matched one of the sequences given as demonstration. However, when the examples had repeated behaviors or behaviors that only appeared in a small percent of the sequences, this was not necessarily the case. When there were repeated behaviors, the number of times the behavior was repeated depended on what amounts the learner could add to meet the post conditions. Furthermore, when a behavior appeared in a small percent of the input sequences, the behavior did not appear in the execution sequence when it was in less than %50 of the input sequences. In addition, as mentioned earlier, during the execution phase the robot can collaborate with a human user, who may help the robot by performing some of the steps in its task. This human intervention may alter the path that the learner would have chosen to perform the task, while still obeying all the constraints of the training samples. Also, if the human user performs a behavior only partially the learner will need to complete it. When a human user performes an incomplete behavior, the number of times a behavior was repeated may not match a seen training example but it will obey the rules represented in the training examples. It is important to note that the human user can perform any behavior

in the execution sequence but the following examples show only the cases where user input had a significant impact on the final execution sequence. In all of the examples the numbers in the parenthesis represent the parameter values for each behavior; the units can be found in the tables below. We tested seven different scenarios in both a simulation and on a physical robot, which cover all of the generalization cases presented in Section III. These examples and the results are presented below.

4.1 Simulation Results

For the simulation experiments, a hand generated text file was given to our system and our system produced the corresponding execution sequence The list of behaviors for the simulation experiments and the parameters that these behaviors can take are shown in Table 4.1.

Table 4.1: Simulation Task Base

Behavior	Abbreviation	Parameters
Add Sugar	`Su`	0.5 cup
Add Salt	`Sa`	1 tsp
Add Baking Soda	`BSo`	1 tsp
Add Flour	`F`	0.5, 1, 2 cup
Add Chocolate Chips	`CC`	0.5, 1, 2 cup
Add Brown Sugar	`BSu`	1, 2 cup
Add Vanilla	`V`	1 tsp
Add Butter	`B`	0.5, 1, 2 cup
Add Eggs	`E`	1, 2 egg
Add Nuts	`N`	0.5, 1, 2 cup
Stir	`St`	1, 2, 3, 4 time

4.1.1 Scenario 1

This constitutes a trivial example, in which all the training sequences were identical, aimed to show that the system can correctly encode tasks from generalization problem 1.1. The following training examples have been provided for this scenario:

- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)
- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)
- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)

Given this input, the system had learned precondition graphs for each behavior. Given that there is no variation in the training examples, the precondition graphs show only one branch for each behavior, representing the only possible path of performing the task. In this case, the execution sequence produced by the system is identical to the training examples:

- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)

If a helper decides to help the learner by performing a behavior in this example, but he does not fully achieve the postconditions of that behavior, then the robot will not produce an execution sequence identical to the training examples. However, the execution will still be consistent with the training examples. In this scenario, if the user interupted the robot to add one egg, the execution sequence looks as follows:

- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (1) E (1) CC (2)

4.1.2 Scenario 2

This scenario aims to illustrate that the system can identify and encode the type of generalizations described in 1.2. The following training examples have been provided for this scenario:

Example 1. In a first setup, the learner is provided with the following demonstrations:

- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)
- CC (2) V (1) F (2) BSu (1) E (2) Su (0.5) B (2) Sa (1) BSo (1)
- BSo (1) E (2) BSu (1) V (1) Su (0.5) Sa (1) CC (2) B (2) F (2)

Figure 4.1 shows the precondition graphs for two of the behaviors, BSo and Su. In this scenario the training examples were all different, with very few ordering constraints that are consistent throughout all the examples (for instance BSu always happens before Su). The system produces an execution sequence similar to one of the training examples (in this case, the third):

- BSo (1) E (2) BSu (1) V (1) Su (0.5) Sa (1) CC (2) B (2) F (2)

If in this example a helper performs a behavior at the beginning of the task, then the robot may take a different path to finish the task. The execution sequence above was choosen at random but if the helper performs the behavior Ffirst, then the following sequence will be executed:

- F (2) Sa (1) BSo (1) B (2) BSu (1) Su (0.5) V (1) E (2) CC (2)

Example 2. In a first setup, the learner is provided with the following demonstrations:

- BSo (1) Sa (1) Su (0.5) CC (2) E (2) F (2) BSu (1) B (2) V (1)
- V (1) F (2) B (2) Su (0.5) E (2) CC (2) Sa (1) BSu (1) BSo (1)
- B (2) E (2) CC (2) Sa (1) BSo (1) BSu (1) F (2) V (1) Su (0.5)
- E (2) F (2) Su (0.5) BSu (1) CC (2) V (1) Sa (1) B (2) BSo (1)
- B (2) E (2) CC (2) BSo (1) Sa (1) Su (0.5) F (2) V (1) BSu (1)

In this scenario the training examples were all different, with no ordering constraints that are consistent throughout all the examples. There were two cases that started with the same three steps and thus the robot choose to start with those three steps since they had occured the most in those positions. At this point in the program, the approach will randomly choose between Sa and BSo. In this example the helper assisted the robot after the first three steps, by performing BSo. Our approach was able to successfully find the next step that obeyed all the rules of the training data.

- B (2) E (2) CC (2) BSo (1) Sa (1) Su (0.5) F (2) V (1) BSu (1)

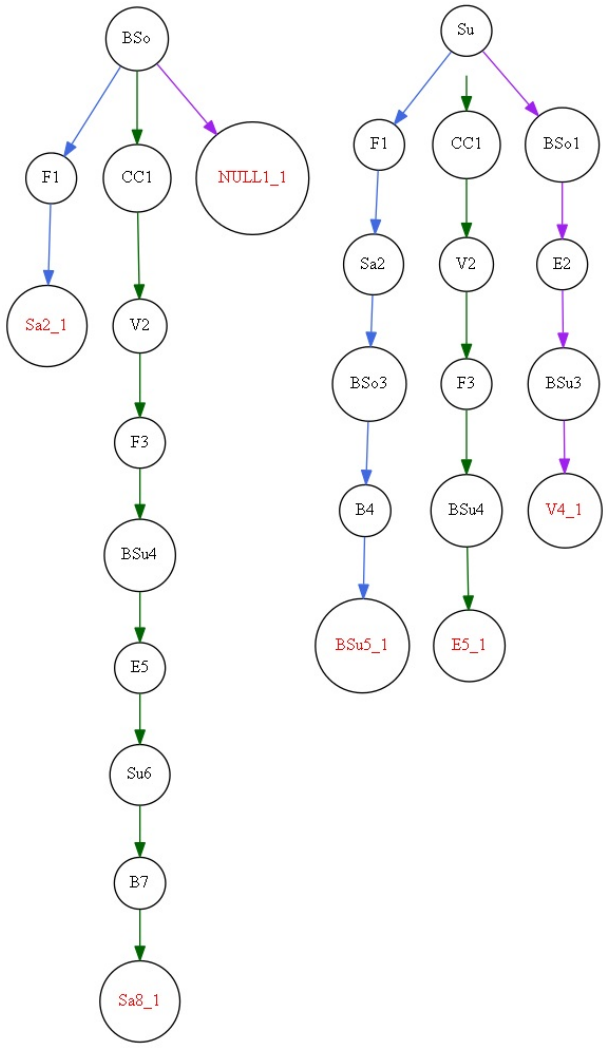

Figure 4.1: Precondition graphs for Scenario 2 (behaviors BSo and Su.)

4.1.3 Scenario 3

This scenario falls under the generalization problem 1.3.1 in which a behavior (or set of behaviors) have to be executed before another behavior (or set of behaviors), but has no ordering constraint with any other behaviors in the task. The following training examples have been provided for this scenario:

- `CC` (2) `Sa` (1) `V` (1) `Su` (0.5) `BSu` (1) `B` (2) `BSo` (1) `E` (2) `F` (2)
- `BSo` (1) `F` (2) `BSu` (1) `Sa` (1) `CC` (2) `E` (2) `B` (2) `Su` (0.5) `V` (1)

- Su (0.5) V (1) B (2) CC (2) Sa (1) F (2) BSu (1) E (2) BSo (1)

Figure 4.2 shows the precondition graphs for two of the behaviors, Sa and E. The above scenario is representative of cases in which a behavior needs to be executed before another behavior but not necessarily immediately before. In the training sequences above, add salt (Sa) always came before add eggs (E) but never right before it. Assuming no behavior has been executed by an outside helper, the system produces an execution sequence similar to one of the training examples (in this case, the first):

- CC (2) Sa (1) V (1) Su (0.5) BSu (1) B (2) BSo (1) E (2) F (2)

As in the last example, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task. The execution sequence above was choosen at random but if the helper performs the behavior BSo first, then the following sequence will be executed:

- BSo (1) F (2) BSu (1) Sa (1) CC (2) E (2) B (2) Su (0.5) V (1)

4.1.4 Scenario 4

This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.3.2, in which a sequence of steps has to be executed in strict consecutive order, but has no ordering constraint with the other step in the task. The following training examples have been provided for this scenario:

- CC (2) F (2) Sa (1) BSo (1) BSu (1) B (2) E (2) V (1) Su (0.5)
- F (2) Sa (1) BSo (1) BSu (1) Su (0.5) CC (2) B (2) E (2) V (1)
- B (2) E (2) V (1) Su (0.5) CC (2) F (2) Sa (1) BSo (1) BSu (1)

In this set of training examples, F always came before Sa which always came before BSo soda and BSo always came before BSu. Also B always came before E which always

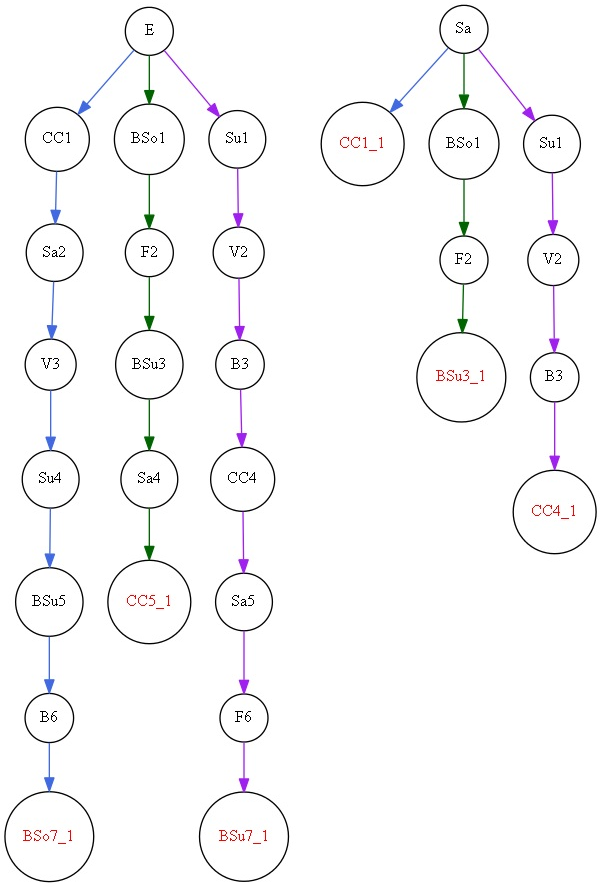

Figure 4.2: Precondition graphs for Scenario 3 (behaviors Sa and E.)

came before `V`. Figure 4.3 shows the precondition graphs for three of the behaviors, two of which are generalized behaviors, `B+E+V`, `F+Sa+BSo+BSu`, and `CC`, which show this general task information is encoded. Assuming no behavior has been executed by an outside helper, the system produces an execution sequence similar to one of the training examples (in this case, the third):

- `B` (2) `E` (2) `V` (1) `Su` (0.5) `CC` (2) `F` (2) `Sa` (1) `BSo` (1) `BSu` (1)

Again, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task and a helper could also perform an incomplete behavior. In the following execution sequence, the external helper performed the behavior `CC` first and then the behavior `B` with only one cup:

- `CC` (2) `F` (2) `Sa` (1) `BSo` (1) `BSu` (1) `B` (1) `B` (1) `E` (2) `V` (1) `Su` (0.5)

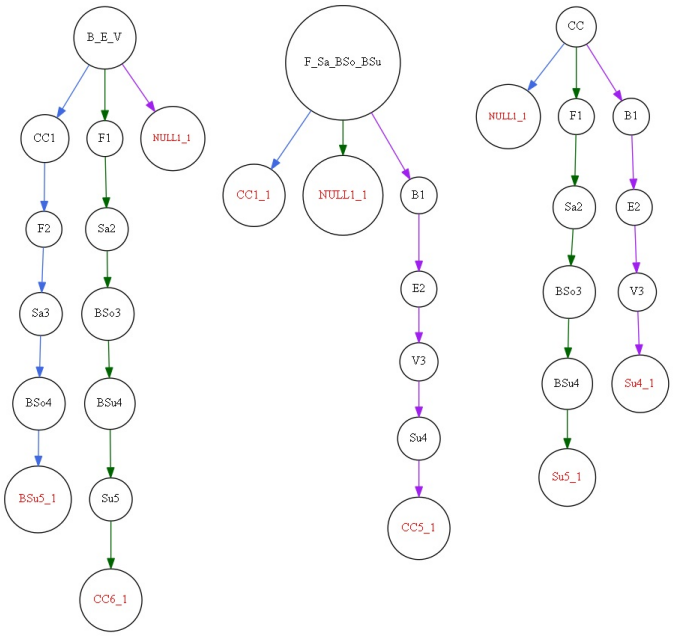

Figure 4.3: Precondition graphs for Scenario 4 (behaviors B+E+V, F+Sa+BSo+BSu, and CC)

4.1.5 Scenario 5

This scenario aims to demonstrate that the system can successfully generalize problems of category 2, in which demonstrations may contain steps that are not relevant and bring no contribution to the task.

Example 1. In a first setup, the learner is provided with the following demonstrations:

- `E` (2) `B` (2) `Su` (0.5) `BSu` (1) `F` (2) `BSo` (1) `N` (2) `Sa` (1) `V` (1) `CC` (2)
- `Sa` (1) `Su` (0.5) `E` (2) `CC` (2) `BSo` (1) `F` (2) `V` (1) `B` (2) `BSu` (1)
- `BSo` (1) `BSu` (1) `B` (2) `CC` (2) `E`(2) `F` (2) `Sa` (1) `Su` (0.5) `V` (1)

In this example, behavior `N` occurs in only one of the three sequences, for an occurence rate of 33%, which is lower than the set threshold of 50% we selected. Therefore `N`'s graph is removed from the set of possible behaviors and is not present in the execution sequence produced by the system:

- `E` (2) `B` (2) `Su` (0.5) `BSu` (1) `F` (2) `BSo` (1) `Sa` (1) `V` (1) `CC`(2)

Moreover, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task. In the following execution sequence, the external helper performed the behavior `BSo` first:

- `BSo` (1) `BSu` (1) `B` (2) `CC` (2) `E`(2) `F` (2) `Sa` (1) `Su` (0.5) `V` (1)

Example 2. In a second setup, the learner is provided with the following demonstrations:

- `E` (2) `B` (2) `Su` (0.5) `BSu` (1) `F` (2) `BSo` (1) `N` (2) `Sa` (1) `V` (1) `CC` (2)
- `Sa` (1) `Su` (0.5) `E` (2) `CC` (2) `BSo` (1) `F` (2) `V` (1) `B` (2) `BSu` (1)
- `BSo` (1) `N` (2) `BSu` (1) `B` (2) `CC` (2) `E` (2) `F` (2) `Sa` (1) `Su` (0.5) `V` (1)

In this example, behavior `N` appeared in two of the three training examples, for an occurence rate of 66% and thus it is included in the generalized representation, as shown by the execution sequence being generated:

- `ABSo` (1) `N` (2) `BSu` (1) `B` (2) `CC` (2) `E` (2) `F` (2) `Sa` (1) `Su` (0.5) `V` (1)

Once again, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task. In the following execution sequence, the external helper performed the behavior `Sa` first and since this training sequence doesn't have nuts in it, the executing sequence won't have nuts either:

- `Sa` (1) `Su` (0.5) `E` (2) `CC` (2) `BSo` (1) `F` (2) `V` (1) `B` (2) `BSu` (1)

Figure 4.4 shows the precondition trees for behavior `N` in both examples.

4.1.6 Scenario 6

This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.3.3, in which a steps has to be executed at the same position (in the total ordering of steps). The following training examples have been provided for this scenario:

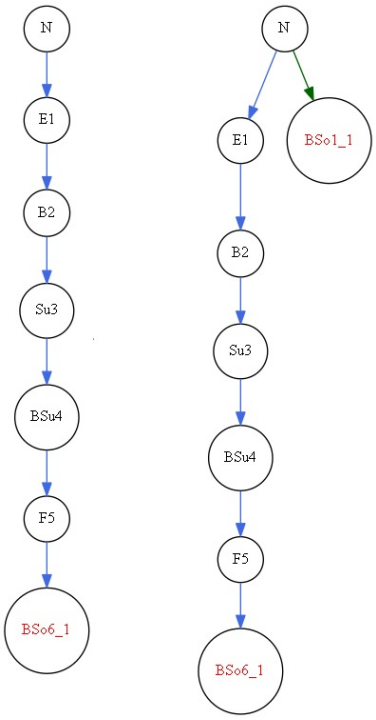

Figure 4.4: Precondition graphs for Scenario 5 (behavior N in examples 1 - left and 2-right)

- CC (2) BSu (1) BSo (1) F (2) B (2) Su (0.5) Sa (1) V (1) E (2)
- V (1) CC (2) F (2) Sa (1) B (2) BSu (1) E (2) Su (0.5) BSo (1)
- Sa (1) Su (0.5) E (2) V (1) B (2) BSo (1) F (2) BSu (1) CC (2)

The above examples are characteristic of a scenario in which a particular aspect of the task needs to happen at the same point in time for every sequence. In this particular case, the behavior add butter (B) was executed fourth in each sequence in the example. As shown in Figure 4.5, the generalized representation captured this aspect of the task, producing an execution sequence consistent with the training examples:

- Sa (1) Su (0.5) E (2) V (1) B (2) BSo (1) F (2) BSu (1) CC (2)

Once more, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task. The execution sequence above was

choosen at random but if the helper performs the behavior `V` first, then the following sequence will be executed:

- `V` (1) `CC` (2) `F` (2) `Sa` (1) `B` (2) `BSu` (1) `E` (2) `Su` (0.5) `BSo` (1)

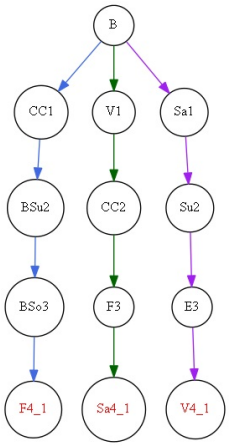

Figure 4.5: Precondition graph for Scenario 6 (behavior B)

4.1.7 Scenario 7

This scenario aims to demonstrate that the system can successfully generalize problems of category 3, in which a particular step (or sequence of steps) may need to be repeated until a certain condition is met. This is illustrated with two examples.

Example 1. In the first setup, the learner is provided with the following demonstrations:

- `Su` (0.5) `Sa` (1) `BSo` (1) `F` (2) `CC` (2) `St` (1) `St` (2) `St` (1) `St` (1) `BSu` (1) `V` (1) `B` (2) `E` (2) `St` (3) `St` (2)
- `BSu` (1) `BSo` (1) `E` (2) `Sa` (1) `CC` (2) `St` (1) `St` (1) `St` (1) `St` (1) `St` (1) `V` (1) `F` (2) `B` (2) `Su` (0.5) `St` (4) `St` (1)
- `B` (2) `V` (1) `E` (2) `BSu` (1) `Su` (0.5) `St` (2) `St` (2) `St` (1) `F` (2) `BSo` (1) `CC` (2) `Sa` (1) `St` (2) `St` (2) `St` (1)

In this case, behavior `St` was repeated five times in two different places in each sequence. Figure 4.6 shows the precondition graph for the repeatable behavior `St`. The system detected the repeated behavior and at execution time it produced a sequence of `St` that used the most efficient way to stir 5 times:

- `BSu` (1) `BSo` (1) `E` (2) `Sa` (1) `CC` (2) `St` (4) `St` (1) `V` (1) `F` (2) `B` (2) `Su` (0.5) `St` (4) `St` (1)

Moreover, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task and a helper could also perform the repeated behavior an incomplete number of times. In the following execution sequence, the external helper performed the behavior `B` first and then the behavior `St` with only two times at the first time it appeared in the example:

- `B` (2) `V` (1) `E` (2) `BSu` (1) `Su` (0.5) `St` (2) `St` (3) `F` (2) `BSo` (1) `CC` (2) `Sa` (1) `St` (4) `St` (1)

Example 2. In the second setup, the learner is provided with the following demonstrations:

- `Su` (0.5) `F` (1) `F` (1) `F` (0.5) `BSu` (1) `Sa` (1) `B` (2) `V` (1) `E` (2) `BSo` (1) `CC` (2)
- `E` (2) `BSo` (1) `Sa` (1) `B` (2) `F` (1) `F` (0.5) `F` (0.5) `F` (0.5) `BSu` (1) `V` (1) `CC` (2) `Su` (0.5)
- `CC` (2) `V` (1) `BSu` (1) `BSo` (1) `Su`(0.5) `E` (2) `F` (1) `F` (1) `F` (0.5) `Sa` (1)

The second example is similar to the first, however instead of one behavior being repeated in the sequence in multiple different places, the behavior `F` was repeated until there was 2.5 cups added in each sequence. The system detected the repeated behaviors and at execution time, produced a sequence whose post conditions met the training examples.

- E (2) BSo (1) Sa (1) B (2) F (2) F (0.5) BSu (1) V (1) CC (2) Su (0.5)

Furthermore, a helper could perform a behavior at the beginning of the task making the robot take a different path to finish the task and a helper could also perform the repeated behavior an incomplete number of times. In the following execution sequence, the external helper performed the behavior Su first and then the behavior F adding only one cup:

- Su (0.5) F (1) F (1) F (0.5) BSu (1) Sa (1) B (2) V (1) E (2) BSo (1) CC (2)

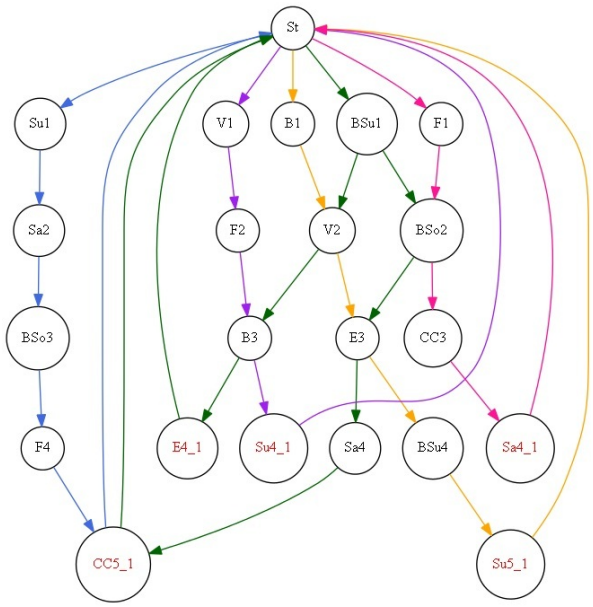

Figure 4.6: Precondition graph for Scenario 7 (Example 1: behavior St)

4.2 Physical Robot Results

The physical robot experiments were carried out using a NAO humanoid robot and a Kinect program as shown in Figure 4.7. The robot's "table" consisted of four colored bowls that were within arm's reach and the robot held onto wooden spoons of equal size. The computer vision portion used the Kinect to segement the teacher's hand out

of the image. The location of the hand was then used to determine what the teacher was interacting with and thus the system was able to extract the behavior associated with the interation. The behaviors were collected as the teacher demonstrated the task and fed into our main system. The graph's were then generated and the execution portion started. When a behavior to be executed was found, our main system sent the behavior to the robot who then carried it out. All of the scenarios below were video recorded and can be found at `www.cse.unr.edu/~monica`. Due to limitations of the robot, we were only able to recreate scenarios 1, 2, 3, 4, 6, and 7 from the simulation section and some of the training examples do not make logic sense (stirring chocolate chips) but are used to demonstrate that we can account for the generalizations found in Section 3.1.

Figure 4.7: Physical robot environmental setup.

The list of behaviors for the robot experiments and the parameters that these behaviors can take are shown in Table 4.2.

Table 4.2: Robot Task Base

Behavior	Abbreviation	Parameters
Add Sugar	`Su`	1 cup
Add Flour	`F`	1 cup
Add Chocolate Chips	`CC`	1 cup
Stir	`St`	1, 2, 3, 4 time(s)

4.2.1 Scenario 1

As in the simulation experiments, this scenario aimed to show that the system can correctly encode tasks from generalization problem 1.1. The following training examples were captured for this scenario:

- `F` (1) `Su` (1) `CC` (1) `St` (1)
- `F` (1) `Su` (1) `CC` (1) `St` (1)
- `F` (1) `Su` (1) `CC` (1) `St` (1)

In this case, all of the training examples are the same so the execution sequence produced by the system is identical to the training examples:

- `F` (1) `Su` (1) `CC` (1) `St` (1)

We ran this scenario a second time and during the second run, a teacher helped out the robot by adding the chocolate chips as the third ingredient. Since the robot no long needs to add the chocolate chips, the execution sequence excludes `CC`. In this scenario, the robot executed the following sequence:

- `F` (1) `Su` (1) `St` (1)

4.2.2 Scenario 2

This scenario aims to illustrate that the system can identify and encode the type of generalizations described in 1.2. The following training examples were captured for this scenario:

Example 1. In a first setup, the learner is provided with the following demonstrations:

- `F` (1) `Su` (1) `CC` (1) `St` (1)
- `Su` (1) `F` (1) `St` (1) `CC` (1)
- `CC` (1) `St` (1) `F` (1) `Su` (1)

Figure 4.8 shows the precondition graphs for the behaviors. In this scenario the training examples were all different, with very few ordering constraints. The robot executed the following sequence which is similar to one of the training examples (in this case, the first):

- `F` (1) `Su` (1) `CC` (1) `St` (1)

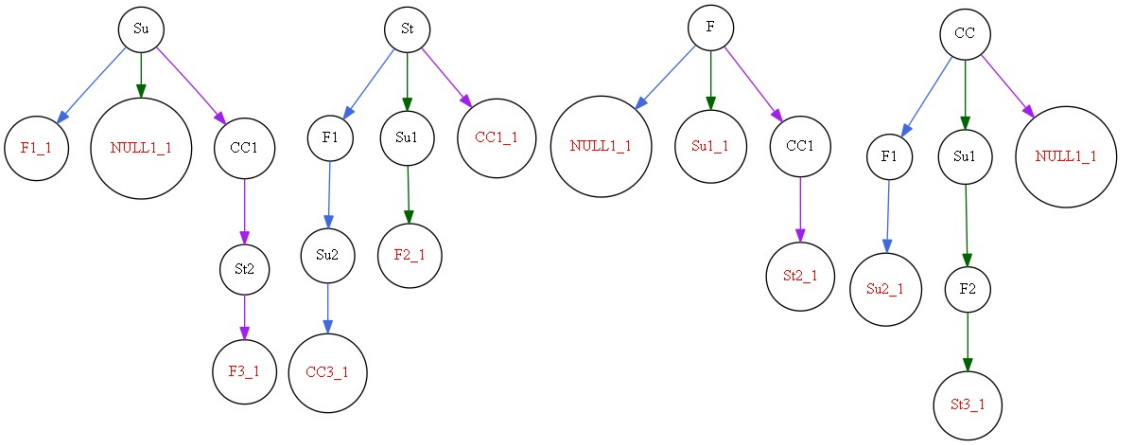

Figure 4.8: Precondition graphs for Scenario 2 Example 1

Example 2. In a first setup, the learner is provided with the following demonstrations:

- `F` (1) `Su` (1) `CC` (1) `St` (1)
- `Su` (1) `F` (1) `St` (1) `CC` (1)
- `CC` (1) `St` (1) `F` (1) `Su` (1)
- `Su` (1) `F` (1) `CC` (1) `St` (1)

Figure 4.9 shows the precondition graphs for the behaviors. Again this example shows that if the same behavior is starting behavior two or more times, then the execution sequence will start with that behavior. The robot executed the following sequence which is similar to one of the training examples (in this case, the second):

- `Su` (1) `F` (1) `St` (1) `CC` (1)

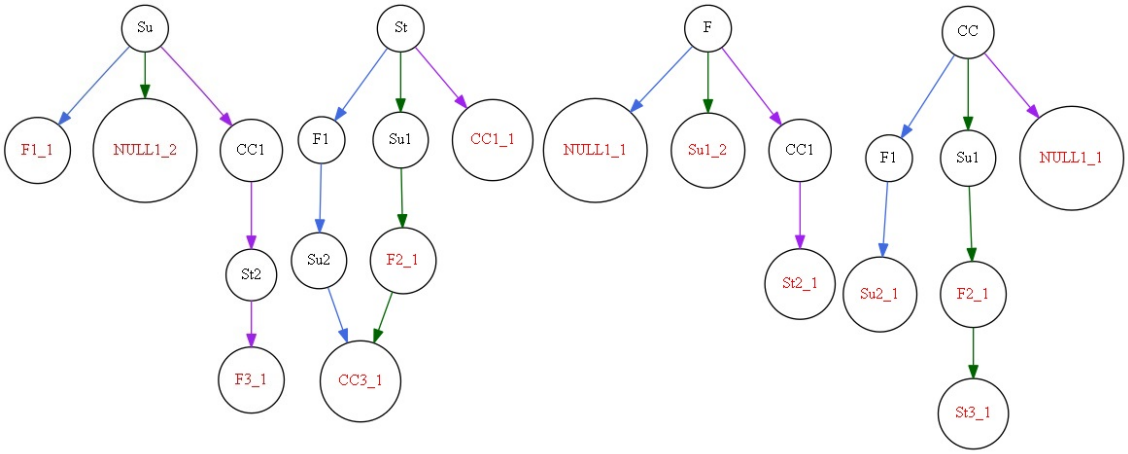

Figure 4.9: Precondition graphs for Scenario 2 Example 2

4.2.3 Scenario 3

This scenario falls under the generalization problem 1.3.1. The following training examples were captured for this scenario:

- `F` (1) `Su` (1) `St` (1) `CC` (1)
- `Su` (1) `St` (1) `CC` (1) `F` (1)
- `Su` (1) `F` (1) `St` (1) `CC` (1)

Figure 4.10 shows the precondition graphs for the behaviors. In the training sequences above, add sugar (`Su`) always came before add chocolate chips (`CC`) but never right before it. The robot executed the following sequence which is similar to one of the training examples (in this case, the second):

- `Su` (1) `St` (1) `CC` (1) `F` (1)

4.2.4 Scenario 4

This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.3.2. The following training examples were captured for this scenario:

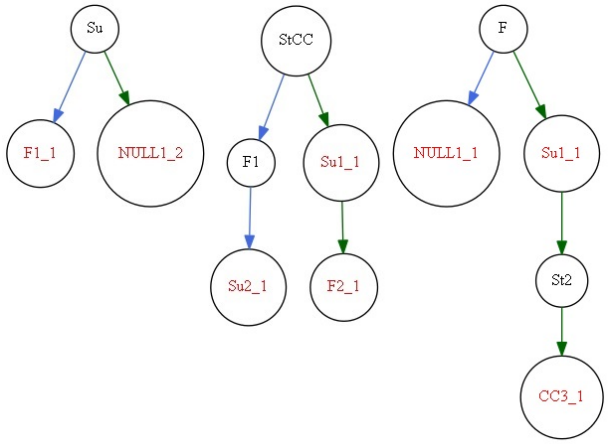

Figure 4.10: Precondition graphs for Scenario 3

- `Su` (1) `F` (1) `St` (1) `CC` (1)
- `CC` (1) `Su` (1) `F` (1) `St` (1)
- `CC` (1) `St` (1) `Su` (1) `F` (1)

In this set of training examples, `Su` always came before `F`. Figure 4.11 shows the precondition graphs for the behaviors, one of which is generalized behaviors. The robot executed the following sequence which is similar to one of the training examples (in this case, the second):

- `CC` (1) `Su` (1) `F` (1) `St` (1)

4.2.5 Scenario 6

This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.3.3. The following training examples were captured for this scenario:

- `CC` (1) `F` (1) `St` (1) `Su` (1)
- `Su` (1) `CC` (1) `St` (1) `F` (1)
- `F` (1) `Su` (1) `St` (1) `CC` (1)

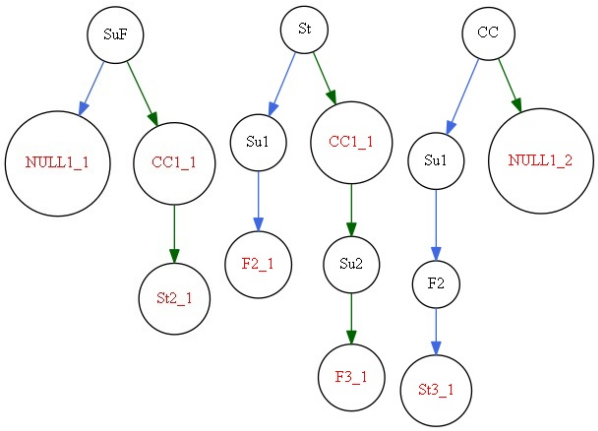

Figure 4.11: Precondition graphs for Scenario 4

In the above examples the behavior stir (`St`) was executed third in each sequence. As shown in Figure 4.12, the generalized representation captured this aspect of the task. The robot executed the following sequence which is similar to one of the training examples

- `Su` (1) `CC` (1) `St` (1) `F` (1)

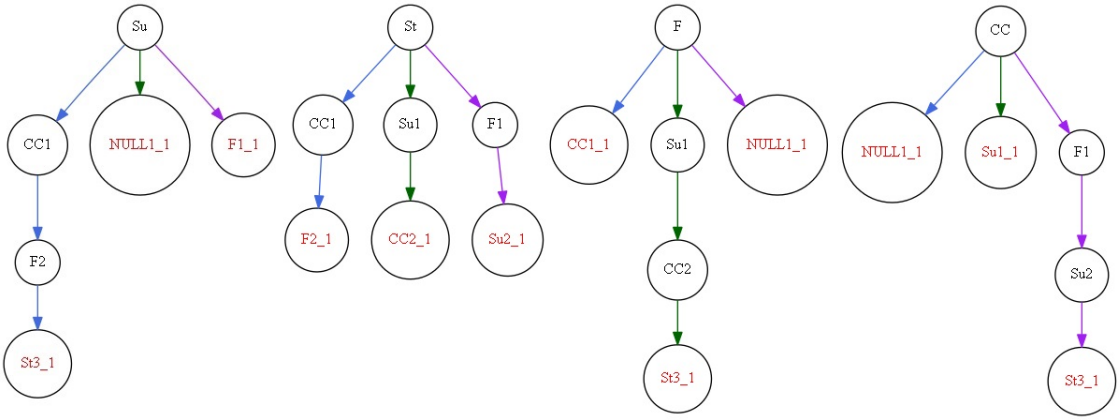

Figure 4.12: Precondition graphs for Scenario 6

4.2.6 Scenario 7

This scenario aims to demonstrate that the system can successfully generalize problems of category 3. The following training examples were captured for this scenario:

- `F` (1) `Su` (1) `St` (1) `St` (1) `St` (1) `St` (1) `CC` (1)
- `Su` (1) `St` (1) `St` (1) `St` (1) `St` (1) `F` (1) `CC` (1)
- `CC` (1) `F` (1) `Su` (1) `St` (1) `St` (1) `St` (1) `St` (1)

In this case, behavior `St` was repeated four times. Figure 4.13 shows the precondition graphs for this scenario. The system detected the repeated behavior and at execution time it produced a sequence of `St` that used the most efficient way to stir 4 times:

- `F` (1) `Su` (1) `St` (4) `CC` (1)

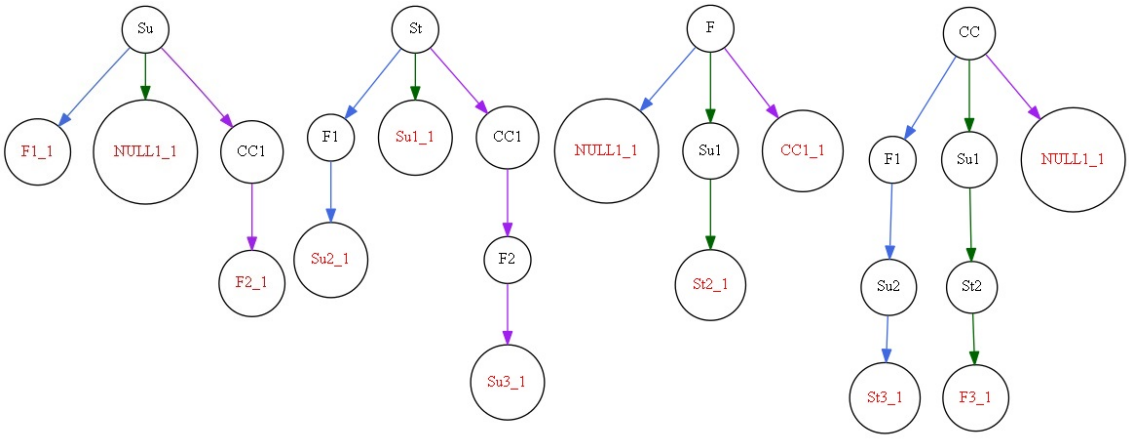

Figure 4.13: Precondition graphs for Scenario 7

Chapter 5

Discussion And Future Work

The experimental results presented above demonstrate that our system can successfully generalize the problems presented in Section 3.1. Furthermore the experiments establish that our system can successfully generate an execution sequence that is always consistent with the training samples. The precondition graphs effectively store a behavior's preconditions as encountered during all demonstrations, thus guaranteeing the system will never produce an execution sequence that breaks any of the ordering constraints encountered in the training examples. The above system has also been successfully tested on a NAO humanoid robot.

Due to the limitations of the NAO robot, we were only able to use one type of spoon. To increase the complexity of the exeperiments performed on the robot, it would benifical to find different various size measuring cups that the robot can use. Furthermore, the NAO robot's left leg was broken and because of this we could only use 4 bowls that were within the robot's reaching area. Fixing the robot's left leg would allow us to add more bowls and thus more complex experiments could be performed. Also our system could create a better execution sequence for the current environment if the state of the environment was recorded before a behavior was executed. This information would be used to choose a behavior in the case where two or more behaviors qualify for the next behavior in the sequence.

Another future direction currently investigated is the ability to enable additional types of generalizations. A first such example would be learning generalizations at the level of a tasks parameters, of tasks such as "Clean up the **blue** or **yellow** toy',

or "Push this chair **10cm** away". Another class of generalization techniques would be on learning tasks with universal quantifiers, of tasks such as "Put **all** the yellow toys in the box and put **all** the other toys on the shelf," or "Throw in the garbage can **everything** that is trash." These present significant challenges, as the robot learner needs to understand that a particular action is desired for all the objects of a particular category. A third clas of techniques would be on learning of conditional, repetitive tasks, of tasks such as: "Turn the knob **until** you hear a click", or "Move to the kitchen **while** holding the tray".

Chapter 6

Conclusion

This thesis proposed a method for teaching a learning robot general task knowledge by generalizing from multiple training samples of the same task. The aim was to extract a task representation that encoded the essential information from all the training examples, in the presence of small or even large variation in the training examples. This was achieved using a representation of the training sequences in the form of precondition graphs, which encapsulated the preconditions of each behavior in the task. These generalized structures were then used to produce an execution sequence that achieved the task at hand and was consistent with the training examples. The method was validated with examples in the task domain of baking cookies both in simulation and on a physical robot. Currently this work is being extended to include additional generalization scenarios and improve the physical robot experiments.

Bibliography

[1] T. Abbas and B. A. MacDonald. Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In *Proc., IEEE Intl. Conf. on Robotics and Automation*, pages 3816–3821, Shanghai, China, 2011.

[2] R. H. Angros. *Learning What to Instruct: Acquiring Knowledge from Demonstrations and foccused experimentation*. PhD thesis, University of Southern California, May 2000.

[3] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proc., Intl. Joint Conf. on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[4] K. Browne and M. N. Nicolescu. Learning to generalize from demonstrations. *Cybernetics and Information Technologies*, 12(3):27–38, 2012.

[5] S. Chernova and M. Veloso. M.: Interactive policy learning through confidence-based autonomy. *J. Artificial Intelligence Research*, pages 1–25, 2009.

[6] S. Chernova and M. Veloso. Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics*, 2(2):195–215, 2010.

[7] G. Fuellen. Multiple alignment. *Complexity International*, 4, 1997.

[8] P. Gaussier, S. Moga, J. Banquet, and M. Quoy. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. Tech Report FS-97-02, AAAI Fall Symp., 1997.

[9] G. E. Hovland, P. Sikka, and B. J. McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Proc., Intl. Conf. on Robotics and Automation*, pages 2706–2711, Minneapolis, MN, 1996.

[10] L. Hugues and A. Drogoul. Synthesis of robot's behaviors from few examples. In *Proc., IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 909–914, Lausanne, Switzerland, 2002.

[11] K. Ikeuchi, M. Kawade, and T. Suehiro. Assembly task recognition with planar, curved and mechanical contacts. In *Proc., IEEE Intl. Conf. on Robotics and Automation*, pages 688–694, Atlanta, Georgia, USA, 1993.

[12] K. Ikeuchi and T. Suehiro. Towards an assembly plan from observation. In *Proc. of Intl. Conf. on Robotics and Automation*, pages 2171–2177, Nice, France, 1992.

[13] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The Internation Journal of Robotics Research*, 31:360–375, 2012.

[14] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transaction on Robotics and Automation*, 10(6):799–822, Dec 1994.

[15] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *Proc., Intl. Joint Conf. on Artificial Intelligence*, pages 1600–1609, Washington, DC, 1993.

[16] A. Larson and R. Voyles. Automatic training data selection for sensory motor primitives. In *Proc., IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 871–876, Wailea, Hawaii, Oct-Nov 2001.

[17] A. Lockerd and C. Breazeal. Tutelage and socially guided robot learning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2004.

[18] A. N. Meltzoff and M. K. Moore. Explaining facial imitation: a theoretical model. *Early Development and Parenting*, 6:179–192, 1997.

[19] H. Miyamoto and M. Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11:1331–1344, 1998.

[20] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. A kendama learning robot based on bi-directional theory. *Neural Networks*, 9:1281–1302, 1996.

[21] A. Munro, M. C. Johnson, Q. A. Pizzini, D. S. Surmon, D. M. Towne, and J. L. Wogulis. Authoring simulation-centered tutors with rides. *International Journal of Artificial Intelligence in Education*, 8:284–316, 1997.

[22] M. Nicolescu, O. C. Jenkins, A. Olenderski, and E. Fritzinger. Learning behavior fusion from demonstration. *Interaction Studies Journal, Special Issue on Robot and Human Interactive Communication*, to appear, 2007.

[23] M. N. Nicolescu and M. J. Matarić. Learning and interacting in human-robot domain. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, Special Issue on Socially Intelligent Agents - The Human in the Loop*, 31(5):419–430, 2001.

[24] M. N. Nicolescu and M. J. Matarić. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Proc., Second Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, July 2003.

[25] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Generation of a task model by integrating multiple observations of human demonstrations. In *Proc., Intl. Conf. on Robotics and Automation*, pages 1545–1550, Washington, DC, 2002.

[26] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Modeling manipulation interactions by hidden markov models. In *Proc., Intl. Conf. on Intelligent Robots and Systems*, pages 1096–1101, Washington, DC, 2002.

[27] D. A. Pomerleau. Rapidly adapting artificial neural networks for autonomous navigation. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 429–435. Morgan Kaufmann Publishers, Inc., 1991.

[28] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *Proc., Intl. Conf. on Robotics and Automation*, pages 578–585, Atlanta, Georgia, 1993.

[29] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 6(3):323–242, 1999.

[30] H. Tominaga, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Symbolic representation of trajectories for skill generation. In *Proc., Intl. Conf. on Robotics and Automation*, pages 4077–4082, San Francisco, California, 2000.

[31] R. Voyles and P. Khosla. A multi-agent system for programming robots by human demonstration. *Integrated Computer-Aided Engineering*, 8(1):59–67, 2001.

[32] R. M. Voyles, J. D. Morrow, and P. K. Khosla. Towards gesture-based programming. *Robotics and Autonomous Systems*, 22:361–375, November 1997.

[33] J. Yang, Y. Xu, and C. S. Chen. Human action learning via hidden markov model. *IEEE Trans. on Systems Man and Cybernetics*, 27(1):34–44, 1997.