Learning Task Specific Plans through Sound and Visually Interpretable Demonstrations

Harini Veeraraghavan and Manuela Veloso
Computer Science Department
Carnegie Mellon University
{harini, veloso}@cs.cmu.edu

***Abstract*— Autonomous robots operating in human environments will need to automatically learn to perform new tasks without requiring the implementation of task-specific actions or time-consuming deliberative planning at run-time. In this work, we contribute a demonstration-based approach for teaching a robot task-specific planners involving complex sequential tasks with repetitions. Complexity of tasks results from step repetitions, execution failures and conditionally executing plans. Our demonstration approach uses sound and visually interpretable cues to guide and indicate the various actions and objects to a robot. The robot in turn performs the actions and generalizes its execution into a task-specific planner. We demonstrate the successful plan learning for two different tasks implemented in real-world settings.**

I. Introduction

A majority of tasks in human environments involve repetition of a complex sequence of actions. The computational requirements of AI planners limits their use in robotics especially where the actions are non-deterministic. Furthermore, the generality of actions with respect to a task can make it challenging for a planner to correctly sequence the actions. For example, Fig. 1 depicts a robot executing different actions that can be sequenced in different orders for performing a variety of dance sequences. However, with no demonstration, it is virtually impossible for a planner to automatically reveal the correct order of a specific dance sequence.

In this work, we contribute a learning by demonstration approach to learning task-specific planners. A demonstration is the execution of a successful solution plan for a given task consisting of a sequence of actions. We focus on the problem of generalizing the same execution into a task specific plan. A task specific plan makes use of a small set of programming like constructs, namely, *if-statements*, *while-loops*, *when-conditional branches*, and *repeatUntil-conditional loops*. A variety of plans arising from any domain can be represented using the task specific plan. In other words, the plan generalization algorithm is not restricted to any specific domain or plan. Learning repetitions allows the robot to easily scale up to problems with any number of repetitions.

To demonstrate a plan, the teacher indicates the appropriate actions and its relevant objects while the robot performs the same actions in the appropriate order. As the robot performs the actions, the teacher can easily include appropriate recovery steps for any observed action failures. The robot's actions are abstract and generate task specific actions upon instantiation on specific objects as indicated by the teacher. This in turn allows the robot to scale to a variety of related tasks using the same action library.

In this work, we contribute a demonstration-based approach for learning task specific plans for complex tasks with step repetitions, execution failures, and conditional executions. By having the robot perform the actions during the demonstration, the teacher can include appropriate recovery steps for the observed action failures.

This paper is organized as follows. After discussing the related work in Section II, we present the task domains in Section III. We then present our learning method in Section IV, results of plan learning in Section V, and conclude the paper in Section VI.

II. Related Work

Planning based approaches such as in [1], [2], [3], [4] suffer from the computational burden resulting from search during run-time. Learning by demonstration approaches such as in [5], [6], [7], [8], [9] learn generalized plans from an observed demonstration of the task. Although the works in [5], [6], [7] learn sequence of actions, repetitions are restricted to single actions. The work in [9] addresses the problem of learning complex looping plans applied only to simulated domains with non-failing actions. The works in [10], [11] learn sequential plans by generalizing routine behaviors and by acquiring exact plans without any generalization, respectively. With the exception of the works in [4], [7], all the afore-mentioned works ignore non-deterministic actions. The work in [4], [12] addresses non-deterministic domains by planning at run-time or by employing transformational planning with pre-specified transformation rules. On the other hand, the work in [7] resolves non-deterministic outcomes by associating a specific action that allows the robot to recruit a human to help complete the task upon failure. Our work adds to this work by learning conditionally executing plans with repetitions due to execution failures. The work in [11] addresses the problem of learning the conditional executions. Our work complements the work in [11] by generalizing conditional executions. Another important issue in robotics is the ability to scale to a variety of tasks using the same set of actions or behaviors. The work in [13] presents a hierarchical

snow-board gorilla airplane hip roll walk dance

Fig. 1. Example snapshots of dancing actions performed by the robot Task 2.

architecture for representing abstract behaviors that can be instantiated for a variety of problems in the same class of a task. In a similar vein, our work uses abstract actions that instantiate into task-specific actions for different objects and used as planning operators for task-specific plans.

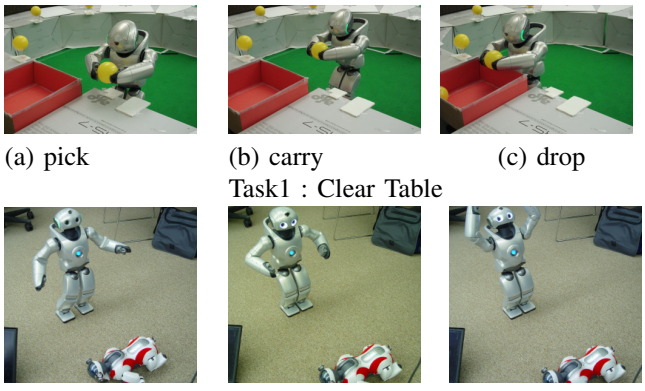
III. Reference Example Learning Tasks

In this work, we make use of two different experimental task domains to demonstrate plan learning. The two tasks are *Clearing the Table* and *Learning to Dance* as depicted in Fig. 2 and described in the following subsections.

A. Task 1: Clearing the Table

The task consists of applying a repeated sequence of actions to move all the objects from a table into a destination box. The robot is assumed to be able to perform some basic actions such as *Search*, *Pick*, *Carry*, *Drop*, etc. However, the robot does not know in what sequence the actions need to performed to achieve the task or the association of appropriate objects to an action. Additionally, actions such as *Pick* and *Carry* non-deterministically result in the robot holding or failing to hold an object at the end of the action. The robot actions descriptions are in Fig. 3.

The main challenge in the task's execution results from the motion of the robot which gives rise to non-deterministic execution of the various actions. For example, unless the robot positions itself at a specific distance and angle with respect to an object, it will fail to pick up the object.



(a) pick (b) carry (c) drop

Task1 : Clear Table

Task 2 : Learn Dance

Fig. 2. The example tasks and the snapshots of actions.

B. Task 2: Learning to Dance

This task consists of performing a specific sequence of actions in response to an observed visual target. Fig. 2 shows snapshots of an action sequence performed by the SONY QRIO robot on observing a visual target (a red colored SONY AIBO robot). The dance step actions do not change the state of the world. Such actions are clearly challenging for a robot to sequence automatically. A sequence of actions are however associated with applicability conditions. Fig. 3 shows an example definition of a robot action for Task 2.

C. Elements of Plan Learning Domain

In this subsection, we define the basic terms used in the learning. The **robot actions** are time extended *abstract robot behaviors* composed of a pre-specified sequence of atomic motions. By using the robot action descriptions and the objects indicated by the teacher during demonstration the robot instantiates the appropriate **task specific action**. The difference between a task specific action and a robot action is that the former is associated with a typed list of arguments, preconditions and effects whereas the latter is more general and applicable to any type of object as depicted in Fig. 3. The preconditions and effects of a task specific action are composed of predicates or propositions where every predicate is a symbolic representation of a visual observation expressing the relation of the objects in the arguments of the predicate. Our earlier work in [14] presents details of transforming the robot actions into task specific actions.

IV. Learning Task Specific Plans through Demonstration

A. Approach Overview

Our approach to learning task specific plans consists of two phases, a demonstration phase and a learning phase. The result of the demonstration phase is an instantiated sequence of the executed task specific actions. The learning algorithm uses the same execution sequence to obtain a generalized task specific plan.

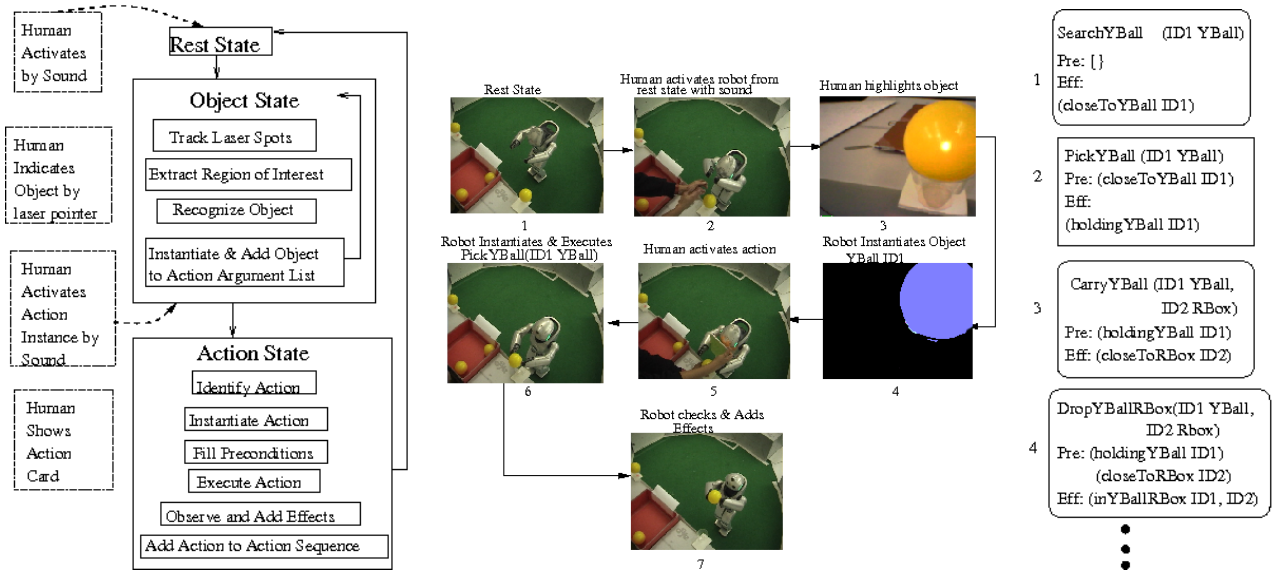
B. Demonstration Phase

The goal of demonstration is to (a) provide the set of task specific actions and the relevant objects, and (b) provide a correct example instantiation of the plan for accomplishing the same task.

The Fig. 4*(A)* depicts the demonstration architecture used in this work. Fig. 4*(B)* depicts the steps in instantiating a single task specific action *PickYBall(ID1 YBall)* through demonstration. As shown, the teacher indicates the various objects and the actions to enable the robot instantiate the task specific actions. Teacher uses sound to transition the

Action name: Pick (Task 1) Arguments: obj – objType Precondition: (closeToObject obj) Effects: (or (holdingObject obj) (failedHolding obj))	Action name: Carry (Task 1) Arguments: obj1, obj2 – objTypes Precondition: (holdingObject obj1) Effects: (and (closeToObject obj2) (or () (failedHolding obj1)))	Action name: Drop (Task 1) Arguments: obj1, obj2 – objTypes Precondition: (and (closeToObject obj2) (holdingObject obj1)) Effects: (in obj1 obj2)
Action name: Search (Task 1) Arguments: obj – objType Preconditions: [ ] Effects: (closeToObject obj)	Action name: Ask (Task 1) Arguments: obj – objType Preconditions: (failedHolding obj) Effects: (closeToObject obj)	Action Name: Soccer Dance (Task 2) Arguments: obj– objType Applicablity Condition : (seeObject obj) Effects: (relevantToGoal)

Fig. 3. Action definitions for Task 1 and Task 2.



(A) Demonstration architecture (B) Task specific action instantiation (C) Action sequence

Fig. 4. The demonstration architecture, an example instantiation of a task specific action, an example instantiated action sequence.

robot to different demonstration modes as well as to move it to appropriate locations in the scene when required. The robot in turn moves by localizing the sound directions and stops on detecting appropriately colored "stop signal" card when displayed by the teacher.

Rest State The rest state (step 1 in Fig. 4*(B)*) precedes every cycle of the task specific action instantiation. Upon sound activation, the robot transitions to the object state as depicted in step 2 in Fig. 4*(B)*.

Object State In the object state, the robot instantiates the objects that belong to the argument list of the current task specific action. Fig. 4*(A)* depicts the various steps involved in this state. The robot obtains the relevant objects by detecting and tracking laser spots as highlighted by the teacher (step 3 in Fig. 4*(B)*) to obtain a region of interest (ROI). The ROI is a bounding box enclosing all the tracked spots. Then using a known model of the object such as color histogram or RGB color thresholds, it identifies the object in the (ROI) as depicted in step 4 in Fig. 4*(B)*. After identifying the object, the robot instantiates the object with an assignment index and stores the object as a object type, object identification pair. The object's type is the symbolic name for the object's model. To consistently identify the object across different actions, the robot maintains an internal store of the object identification which is deleted based on the executed action. For example, an action *dropObject x1 y1*will remove the object x from the internal store.

Action State The teacher enables the transition of the robot from the object state to the action state through sound. Once in the action state, the teacher indicates the action using a colored card as depicted in step 5 in Fig. 4*(B)*). The robot identifies the action by applying color thresholds on the observed image and instantiates the corresponding task specific action. It then computes the appropriate action preconditions by transforming the visual measurements into predicates. Next, the robot executes the action (step 6 in Fig. 4*(B)*), obtains the effect predicates (step 7 in Fig. 4*(B)*), stores the action, and transitions back to *Rest State*. After the demonstration, an entire sequence of instantiated actions as depicted in figure C in Fig. 4 is obtained. The action 2 in figure C of Fig. 4 corresponds to the task specific action depicted in figure B of Fig. 4.

C. Learning Phase

The goal of learning is to produce a generalized task-specific plan for the demonstrated task. Learning proceeds

in two stages. In the first stage, the action sequence obtained from the demonstration is analyzed to extract the partial step ordering constraints using an algorithm as in [15]. The algorithm links steps that share a producer-consumer relationship, where the effect of the producer step is a precondition of the consumer step. Additionally, the algorithm annotates the links with the precondition linking the two steps as depicted in Fig. 5.

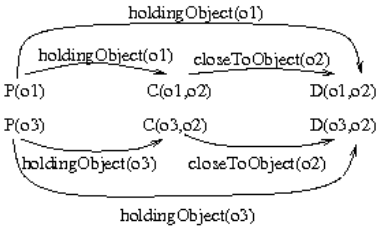


Fig. 5. Annotated partial step orderings of action sequence

In the second stage of learning, the annotated partial step orderings of the demonstration is generalized into a task specific plan. The extracted plan is represented using a domain specific planner language (dsPlanner) similar to the work of [9]. We represent the plan using programming like constructs such as parameterized *If* statements for step preconditions, *While* loops for parallel sequences of a loop, *When* statements for representing the conditions of conditional plans, and *Repeat Until* loops for plan recovery structures with repetitions.

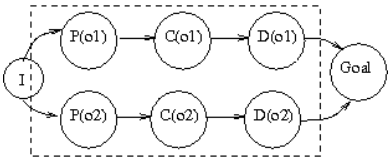


Fig. 6. Example of a loop

Algorithm 1 Learning Task Specific Plans from Example

Input: Partial Order (PO) Graph
Output: Generalized Task Specific Plan
1: Transitively reduce PO Graph
2: Detect LOOPS(Actions $a_1,\ldots,a_N$)
3: COMBINE Conditional Actions
4: COMBINE loops(Loops $L_1,\ldots,L_m$)
5: **For all** Loop and Conditional branches S_b **do**
6: Detect REPEAT UNTIL LOOPS(S_b)
7: MERGE Repeat until loops
8: Order Structures and steps by links.

Loops represent sequences of actions operating on different instances of a loop variable. Hence, the branches of a loop execute in parallel as depicted in Fig. 6. Two actions *opA(o1)*, and *opB(o2)* are said to be in a loop iff the actions *opA, opB* do not have any producer-consumer links and are **equivalent**. Two actions are equivalent ($opA \equiv opB$) when they have the same action names and argument types. The condition for a loop can be expressed as,

$$\begin{cases} opA \equiv opB \wedge LINK(opA,opB) = \emptyset, & then\ True \\ & otherwise\ False \end{cases}$$

Similarly, two loops $L_1(?v_i)$ and $L_2(?v_j)$ can be **combined** into a single loop iff the action instances make use of the same values of the loop variables. In other words,

$$\begin{cases} v_i = v_j, & then\ TRUE \\ & otherwise\ FALSE \end{cases} \quad (1)$$

When condition is associated with a conditionally executing sequence. The condition consists of a conjunction of predicates upon which a sequence of actions is executed. Fig. 7 represents two conditional plans that execute on different conditions $X1$ and $X2$. A conditional plan is obtained by **combining** actions or loops having the same applicability conditions.

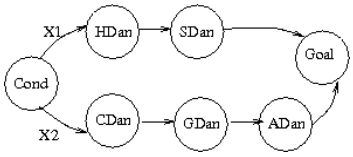


Fig. 7. Example of a conditional plan.

Algorithm 2 Learning Repeat Until Loops

Input: Execution Sequence Seq
Output: All repeat until loops
1: idx $\longleftarrow$ 0
2: **Repeat**
3: $\{idx, opA_t, opA_{t+\Delta}\} \longleftarrow$ Find Non-deterministic action (Seq, idx)
4: **If** $idx <$ Seq length **then**
5: Get Actions $opX = \{opx_{t+1},\ldots,opx_{t+\Delta-1}\}$ in Seq
6: Convert opX into parameterized operators
7: Order operators by ordering constraints and add to repeat until loop
8: $idx \longleftarrow t+\Delta+1$
9: **Until** $idx <$ Seq length

Repeat Until Loops represent repetitions of one or more actions occurring on the same instance of a loop variable such that every repetition of an action has a strict ordering constraint with respect to its previous occurrence in the loop. The repeat until loop is associated with a terminal action that starts and terminates the loop. For example, the action $P(o_1)$ in Fig. 8 is the terminal action of the depicted loop. The terminal action is a non-deterministic action and its effect that terminates the loop is the condition for the repeat until loop. Two repeat until loops can be **merged** when the condition and the body of the loops match exactly. Two bodies match when every action in the body of the two loops are equivalent.

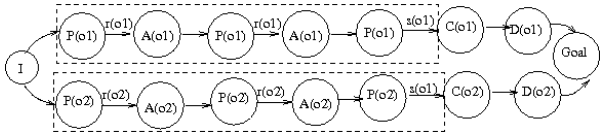


Fig. 8. Example of a repeat until loop.

The algorithm for generalizing the action sequence into a task

specific plan is depicted in Alg. 1. As shown, in the Line 1 of Alg. 1 the partial order graph is first transitively reduced to simplify the computation for plan extraction. In the next step as depicted in Line 2 of Alg.1, loops are detected on the individual actions using the definition as in Eqn. 1. In Line 3 of Alg.1 the actions are combined to obtain the longest sequence of conditional action sequence. The actions are combined by iteratively matching the applicability conditions of the merged condition blocks and single actions. In Line 4 of the Alg.1, the different loops are combined using the definition in Eqn. 1 to obtain the longest looping action sequence. Finally, the repeat until loops are detected from the branches of the loops and conditional plan sequences not associated with any loop using the Alg.2. Individual branches are tested for repeat until loops as the actions in a repeat until loop correspond to the same instance of the loop variable. The repeat until loops are merged in Line 7 of Alg. 1 by exactly matching the conditions and bodies of the loops.

The algorithm for detecting repeat until structures is depicted in Alg. 2. As shown, the algorithm successively searches the sequence for repeat until structures. A repeat until structure is detected using its definition as described earlier by first identifying the terminal action consisting of the earliest occurring non-deterministic action (Line 3 of Alg.2) with at least two occurrences. Lines[5-7] of Alg. 2 depicts the extraction of the non-terminal actions in the repeat until structure. Actions are instantiations of operators and by parameterizing the actions, the algorithm merges the actions into operators in Line 6 of Alg.2.

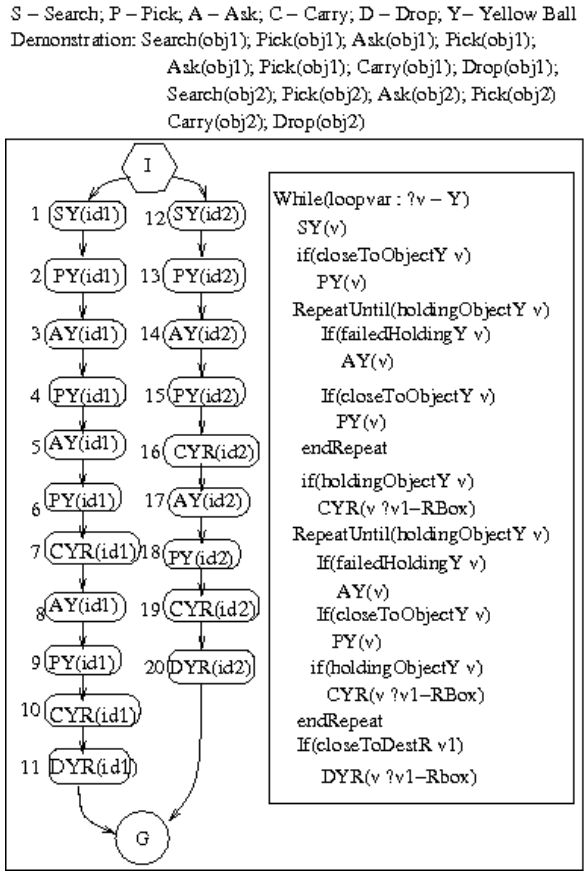


Fig. 9. Task specific plan for a demonstrated solution for Task 1.

V. Plan Learning Experiments and Discussion

The goal of the experiments was to test the efficacy of the learning from demonstration approach. All the demonstration experimental runs were performed in an indoor environment using an experimental setup as depicted in the Fig. 2. The goal of task 1 was to test plan learning for tasks with repetitions and non-deterministic action outcomes. The environment was controlled to limit the amount of uncertainties from vision. Uncertainties arise from both robot motion and visual sensing. An example demonstration run and the corresponding learned plan is depicted in Fig. 9. As shown, the algorithm correctly transforms the robot actions into task specific actions and learns the correct task specific plan. The correctness of the learned plan was verified by manual inspection. Despite different sequence lengths of the repeat until loops, $\{AY(id1);PY(id1);AY(id1);PY(id1)\}$ in the first branch and $\{AY(id2);PY(id2)\}$ in the second branch depicted in Fig. 9, the algorithm correctly merges the two branches into a single repeat until loop.

Fig. 10 depicts the demonstration trace and the generalized plan for the Task 2. As shown, the algorithm correctly identifies the conditional plan and the loop. Fig. 11 depicts the demonstration trace and the generalized plan for the Task 2 for two different conditional plans resulting from two different visual percept.

The presented approach learns correct looping plans as long as the demonstrations include at least two instantiations of the same sequence of actions (instantiated with different loop variables). The approach also learns failure recovery plans thereby making the learned planners robust to some of the non-determinism in the environment. As the learned plans are executable, there is no need to invoke a search based planning algorithm during execution. Also, the ordering of actions in sequence automatically constrain the visual percept that the robot will attend to, thereby, ensuring that the robot executes the task correctly. For example, when performing an action to pick a yellow colored ball, the robot will not even process the visual information corresponding to green colored balls.

Although we present the results of plan learning only for two different domains, the algorithms are general and applicable to a variety of domains. In a different work, we have successfully applied the same plan generalization algorithms to extract plans for scheduling using web services [16]. The dsPlanner language used to represent the plans itself can be specified easily ahead of time. The work that is closest to the presented work is the work in [9] that learns looping plans in domains where actions do not have any failed executions. Our approach extends this work by analyzing plans that have conditional branches as well as failed executions.

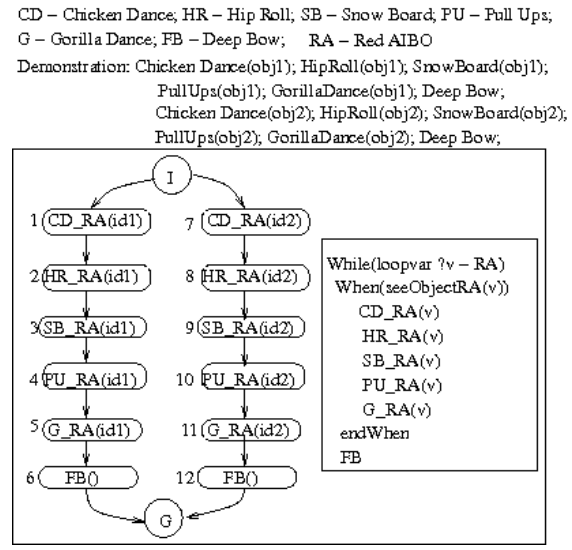


Fig. 10. Task specific plan with loop, conditional branch for Task 2.

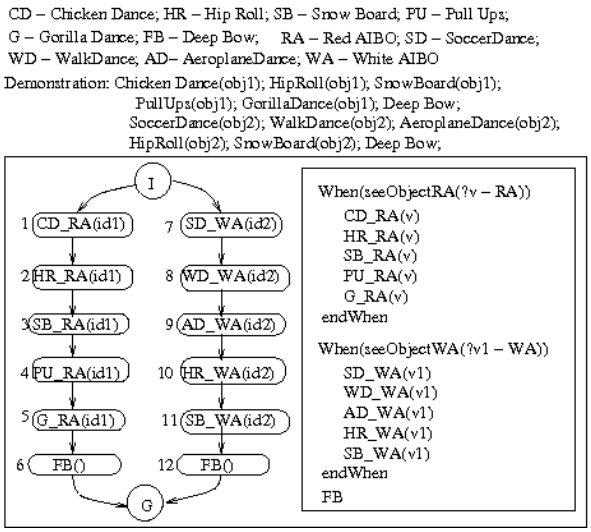


Fig. 11. Task specific plan with conditional branches for Task 2.

VI. Conclusions

In this paper we presented an approach to learning task specific plans from sound and visually interpretable demonstrations. Our demonstration approach makes use of easy to detect visual cues and correctly transforms the observed demonstration into an execution sequence of instantiated task specific actions and successfully generalizes the observed execution sequence into an executable task-specific plan. Our approach has been implemented and tested on two different tasks with different plan structures under different experimental settings and different visual objects.

VII. Acknowledgements

The authors would like to thank SONY corporation for providing the SONY QRIO robots and the robot specific software libraries for this research.

References

[1] N. Nilsson, "Shakey the robot," SRI International, AI Center, SRI International, Menlo Park, CA, Tech. Rep. 323, 1984.

[2] M. Schoppers, "Universal plans for reactive robots in unpredictable environments," in *In Proc. Intl. Joint Conf. AI*, 1987, pp. 1039–1046.

[3] K. Haigh and M. Veloso, "Interleaving planning and execution for asynchrnous user requests," *Autonomous Robots*, vol. 5, no. 1, pp. 79–95, 1998.

[4] M. Beetz, T. Arbuckle, T. Belker, A. Cremers, D. Schulz, M. Bennewitz, W. Burgard, D. Hähnel, D. Fox, and H. Grosskreutz, "Integrated, plan-based control of autonomous robots in human environments," *IEEE Intelligent Systems*, vol. 16, no. 5, pp. 56–65, 2001.

[5] N. Koenig and M. Mataric, "Demonstration-based behavior and task learning," in *Working Notes AAAI Symposium To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: extracting reusable task knowledge from visual observation of human performance," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994.

[7] M. Nicolescu and M. Mataric, *Models and Mechanisms of Immitation and Social Learning in Robots, Humans, and Animals: Behavioral, Social and Communicative Dimensions*, 2006, ch. Task learning through immitation and human-robot interaction, pp. 407–424.

[8] C. Breazeal, G. Hoffman, and A. Lockerd, "Teaching and working with robots as a collaboration," in *Proc. Autonomous Agents and Multiagent Systems*, 2004, pp. 1028–1035.

[9] E. Winner and M. Veloso, "Loopdistill: Learning domain-specific planners from example plans," in *In ICAPS Workshop on Planning and Scheduling*, 2007.

[10] S. Chernova and R. Arkin, "From deliberative to routine behaviors: a cognitively inspired action-selection mechanism for routine behavior capture," *Adaptive Behavior*, vol. 15, no. 2, pp. 199–216, 2007.

[11] P.E.Rybski, K. Yoon, J. Stolarz, and M. Veloso, "Interactive robot task training through dialog and demonstration," in *Proc. Human Robot Interaction Conf.*, 2007.

[12] A. Müller, A. Kirsch, and M. Beetz, "Transformational planning for everyday activity," in *Proc. Intl. Conf. Automated Planning and Scheduling*, 2007, pp. 248–255.

[13] M. Nicolescu and M.J.Matarić, "A hierarchical architecture for behavior-based robots," in *Proc. Intl. Joint Conf. Autonomous Agents and Multi-Agent Systems*, 2002, pp. 227–233.

[14] H. Veeraraghavan and M. Veloso, "Teaching sequential tasks with repetition through demonstration," in *(Short Paper) Proc. Intl. Conf. on Autonomous Agents and Multi-Agent Systems*, Paghadam, Parkes, Muller, and Parsons, Eds., 2008.

[15] E. Winner and M. Veloso, "Analyzing plans with conditional effects," in *Proc. Intl. Conf. Artificial Intelligence and Planning Systems*, 2002, pp. 23–33.

[16] H. Veeraraghavan and M. Veloso, "Learning looping plans for web service composition from example execution of web services," Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA, technical report CMU-CS-08-124, May 2008.