Towards Goal Based Architecture Design for Learning High-Level Representation of Behaviors from Demonstration

Benjamin Fonooni, Thomas Hellström, and Lars-Erik Janlert

Abstract— **This paper gives a brief overview of challenges in designing cognitive architectures for Learning from Demonstration. By investigating features and functionality of some related architectures, we propose a modular architecture particularly suited for sequential learning high-level representations of behaviors. We head towards designing and implementing goal based imitation learning that not only allows the robot to learn necessary conditions for executing particular behaviors, but also to understand the intents of the tutor and reproduce the same behaviors accordingly.**

***Index Terms*— Learning from Demonstration, Cognitive Architecture, Goal Based Imitation**

I. INTRODUCTION

LEARNING from Demonstration (LfD) is one of the most popular learning techniques to teach robots new skills by observing a human or robot tutor [2]. LfD involves several challenges, such as generalization of learned behaviors, representation of behaviors, sequence learning, and reproduction of complex behaviors [23], [24], [25]. Some researches proposed architectures based on biology and psychology of human or animal cognitive systems [3]. Examples of biologically inspired models are given by Billard et al. [7], Kopp et al. [8] and Demiris et al. [19]. The neural model approaches, fundamentally driven by mirror neuron systems [10], are also considered by many researchers [18], [19]. There are also some recent efforts in modeling goal-based imitation that infer intents of the tutor rather than repeating observed actions and following exact trajectories [4], [5], [16].

Most of the works referred above, focus on learning and reproduction of low-level representations (sensory-motor events) of behaviors. In this work we assume that these representations are already available as *behavior primitives* (below often referred to as *primitives*) such that no learning is required at the sensory-motor level. Primitives have been applied in robot control for several years, and there are proposed models describing challenges of connecting perception to primitives [13]. Primitives accomplish goal-directed behaviors and can be formalized as control policies [1]. Primitives may also represent complete temporal behaviors [20], [21].

The main goal of the work presented in this paper is to introduce a novel architecture for learning *contexts,* which are high-level representations of behaviors. Each context is associated with a predefined *action* and contains information on necessary perceptual conditions for this action to be executed. Actions are part of the architecture and act as interfaces between contexts and primitives in order to retrieve objects of attention from the contexts, convert them into low-level information and pass them as parameters to primitives.

We improve the previously developed architecture [6] by implementing cognitive mechanisms to learn intentions of the tutor and reproduce the behavior through activating learned contexts and recognizing the associated stimuli.

The remainder of the paper is structured as follows: In the next section, the proposed architecture and its components are elaborated. Section III introduces a novel algorithm for learning new contexts, and mechanisms for reproduction of learned behaviors. Section IV describes results from several experiments. Section V explains goal inference mechanisms which are key factors for behavior reproduction.

II. ARCHITECTURE OVERVIEW

Fig. 1 depicts the developed architecture. In the following sections all units and modules are described.

A. *Perception Unit*

This unit is responsible for perceiving the environment by processing sensor data. Sensors can deliver either low-level data, like laser scanners, or high-level data, like gesture recognizers, emotion detectors and RFID tag readers. The difference between low and high-level data is the amount of processing required to connect the output of the sensors with concepts. For instance, reading the RFID tag of a cup and fetching its properties from a database, requires less processing than perceiving the cup and its properties solely by a laser scanner.

The Perception Unit delivers processed information to the various modules of the Cognition Unit.

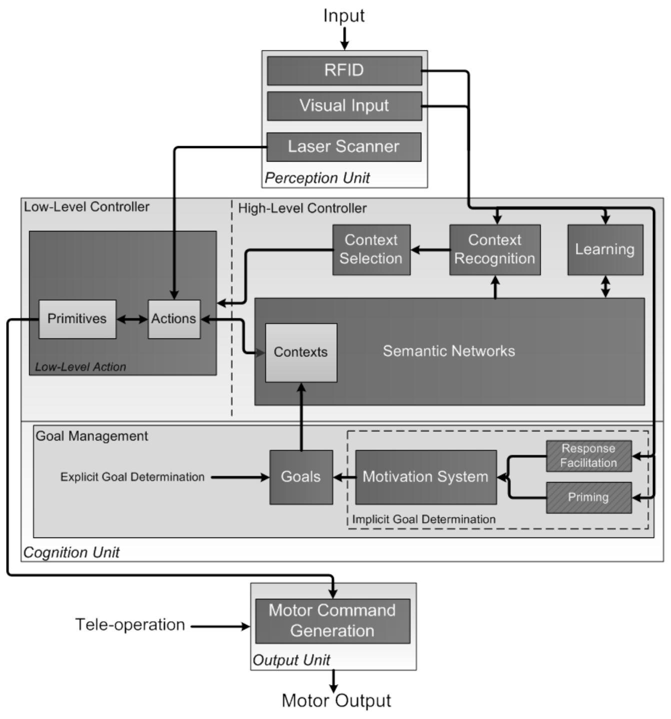

Fig. 1. Units and modules of the architecture

B. *Cognition unit*

The Cognition Unit consists of three main modules. The High and Low-Level controllers are responsible for learning, recognizing and executing contexts. The Goal Management module is designed for creating and inferring goals, explicitly and implicitly, as well as keeping track of current goal and intentions.

High-Level controller

This module is responsible for learning contexts. A main component of this module is the long-term memory represented by an, initially predefined, Semantic Network (SN). It contains nodes representing concepts and objects according to a pre-defined ontology. Each context is a node in the SN. By the learning process, it gets associated with a behavior primitive, and with concepts and objects in the SN. In our previous work we proposed a novel approach that creates connections (links) between the context and nodes connected to perception [6]. Perceptions are outputs of the perception unit, and activate the corresponding nodes in the SN. Each node has an activation level that defines how strongly it is activated.

After the learning phase, the robot should be able to recognize conditions for triggering a specific context, and thereby executing the associated behavior. This is denoted the reproduction phase. The perception mechanisms will change activation levels of sensed nodes, which in turn are connected to one or more contexts. Due to the spreading activation mechanisms in the SN, activation will be propagated to connected nodes [11] such that the robot will be able to generalize the learned contexts stored in the SN [12]. In this way, the Context Recognition module selects one or several contexts.

The most highly activated context will be selected by the Context Selection module and made available to the Low-Level Action Controller.

Low-Level controller

As mentioned earlier, the robot is equipped with a set of pre-defined behavior primitives. In the learning phase, the High-Level Controller associates a behavior primitive with the newly learned context via actions. This association is automatically recognized by the robot during tele-operation. In the reproduction phase, the Low-Level Controller is responsible for selecting motor commands in accordance with the pre-defined scheme in the selected primitive, and passing them to the Output Unit for execution.

a) Primitives

As described above, primitives are pre-defined low-level representations of behaviors. In the examples in this paper, we use “Grip” (gripping an object with the robot arm) and “Go to Location” (moving the robot to a particular location). The associations between actions and primitives are pre-defined.

b) Actions

Actions have an intermediary role in connecting contexts with primitives. They retrieve objects of attention from contexts and convert them to parameters required by the primitives. For instance, if the context “Get the Cup” is activated by perceiving the “Cup 1” as an object of attention, the associated action will pass “Cup 1” as parameter to the “Go to Location” primitive. The concept *Object of attention* refers to an object that the robot is going to work on. In the examples in this paper, we use three actions, each one mapped to a primitive. “Explore and Reach” is mapped to “Go to Location”, “Grab” is mapped to “Grip”, and “Move to Safe Location” is also mapped to “Go to Location”. The reason to keep actions and primitives separated is due to the possible association of several actions to one primitive. This allows actions to have different sets of conditions and way of providing parameters while primitives are only focusing on low-level aspects.

All actions and primitives are pre-defined in the system. The necessary pre-defined actions and primitives depend on the scenario and more importantly, the robot’s capabilities.

Goal Management

For most complex behaviors, several primitives have to be activated in sequence. A goal represents a sequence of contexts. Each goal has a set of conditions and objects of attention that defines what to look for and when to activate a specific context. Such a goal can be *“Help human rescue a victim”*, *“Moving victim to a safe place”* and etc.

One of the advancements of the current design in comparison with the previously developed architecture [6] is the Motivation system. It may be triggered by cognitive mechanisms such as response facilitation and priming, which motivate the robot to choose specific goals and eventually execute desired actions. This is denoted as implicit goal determination. Response facilitation is the phenomenon when observing a specific act, which is already in the repertoire of the robot, increases the probability of the robot later performing the same act [9]. One example is if the robot observes a human approaching a cup and grabbing it. This is identified as “Grab the Cup” behavior and increases the probability of the robot executing the same behavior.

Priming can be defined as an implicit memory effect that speeds up the response to stimuli because of exposure to a certain event or experience [22]. In our case, priming is the pre-activation of concepts stored in SN, in order to bias the learning process or affect the goal selection mechanism. For instance, if a “Red Ball” is shown to the robot, the nodes “Red” and “Ball” are primed and pre-activated. Therefore, the chances to select and satisfy goals that have connections to “Red” or “Ball” increase.

C. Output unit

This unit converts low-level commands from the Cognition unit to motor commands. In addition, it enables tele-operation of the robot.

III. Learning And Reproduction Phases

In the learning phase, a demonstration of a desired behavior is used to associate high-level contexts with perceived information. Each context is also mapped to a behavior primitive via an action such that the primitive will be executed when similar perception occurs during the reproduction phase.

A. Learning

The learning process is one of the main tasks of the architecture. In our previous work we have developed a learning algorithm based on novelty detection technique [6]. In this paper we will describe a new context-learning algorithm called *Multiple Demonstrations* (MD). We assume that we already have a number of predefined primitives and a predefined SN based on an ontology of the domain in which the robot should operate. The SN is interfaced to the Perception unit and activates related nodes through spreading and decaying activation mechanisms [11].

Context creation

The learning process starts by a tutor demonstrating the wanted behavior through tele-operation. A new context node is added to the SN. The robot observes the environment by sampling sensors at a given frequency. In the reported experiments, RFID tags are used for simplified object detection and identification. Each read-out gives identities and properties of objects perceived in the environment and causes the corresponding nodes to be activated. For instance, if the RFID belonging to a red ball is detected, the nodes “Red” and “Ball” will be activated. In this way, the RFID reader emulates sensors for object type and color. Throughout the learning process, activation levels propagate to all connected nodes by spreading activation. This mechanism allows the robot to generalize one concept to another. For controlling the degree of generalization, we define an energy level variable for each node. The energy level of a node determines how far activation level will spread from the initial node [15].

Sometimes nodes are deactivated during the demonstration due to noise and uncertainties in the RFID equipment. Therefore, a decaying delay parameter is defined to prevent immediate deactivation of a node when the corresponding object is not longer perceived in the environment.

The same behavior must be demonstrated to the robot at least twice. A new SN will be created each time, and the context node will be connected to nodes activated by the RFID read-outs. Due to noise and varying external conditions, these nodes may differ between demonstrations. To finalize the learning process, two issues must be solved: First, the most relevant connections must be determined. Second, suitable weights between the remaining nodes and the context node must be computed. In order to identify relevant connections, the MD algorithm looks for nodes with similar activation levels in all demonstrations. One-Way ANOVA [14] is used to compare mean node activation values of all nodes. The null hypothesis is that there is no significant difference between demonstrations for activation of a node. The following computations are performed for each one of the nodes connected to the new context node.

For each demonstration, sum of activations (S_{A_x}), activation mean value (μ_{A_x}), squares of deviations (d^2) and sum of squares of deviations ($S_d{}^2$) are calculated for each node:

$$d^2 = (A_x - \mu_{A_x})^2 \tag{1}$$

where A_x is the activation value of node *x*.

$$S_{d^2} = \sum {A_x}^2 - \frac{(\sum A_x)^2}{n} \tag{2}$$

where n is number of samples in the demonstration.
Grand Total (GT) is calculated as

$$GT = \sum S_{A_x}. \tag{3}$$

Then we calculate *total sum of squares (T)* as

$$T = GT - \frac{(GT)^2}{\sum_{i=1}^{r} n_i} \tag{4}$$

where *r* is the total number of demonstrations and n_i is the number of samples in demonstration *i*. *Between groups sum of squares (BG)* is calculated as

$$BG = \sum_{i=1}^{r} \frac{(S_{A_{x_i}})^2}{n_i} - \frac{(GT)^2}{\sum_{i=1}^{r} n_i}. \tag{5}$$

W*ithin groups sum of squares (WG)* is calculated as

$$WG = GT - \sum_{i=1}^{r} \frac{(S_{A_{x_i}})^2}{n_i}. \tag{6}$$

The number of degrees of freedom for *between groups sum of squares (BDF)* is calculated as

$$BDF = r - 1. \tag{7}$$

The number of degrees of freedom for *within groups sum of squares (WDF)* is calculated as follows:

$$WDF = r\left(\frac{\sum_{i=1}^{r} n_i}{r} - 1\right). \quad (8)$$

The *total degree of freedom (TDF)* is calculated as follows:

$$TDF = WDF + BDF. \quad (9)$$

Finally the F value is calculated as

$$F = \frac{\frac{BG}{BDF}}{\frac{WG}{WDF}}. \quad (10)$$

The *F* distribution (*p*=0.05) with the given *BDF* and *WDF* is then looked up. If the calculated *F* has higher value, we reject the null hypothesis and conclude that there is a significant difference between demonstrations for activation of the node. The node is then disconnected from the context node. The process is repeated for all nodes initially connected to the context node. Finally, weight values for the remaining nodes are calculated as

$$w_x = \frac{\sum_{i=1}^{r} N_{x_i}\mu_{A_{x_i}}}{P} \quad (11)$$

where N_{x_i} is the number of samples for which node *x* has activation value above 0 during the *i*th demonstration, and *P* is the weighted sum for all nodes, calculated as

$$P = \sum_{i=1}^{r}\sum_{j=1}^{n} N_{j_i}\mu_{A_{x_{j_i}}}. \quad (12)$$

After the process of context forming, the goals are created and related contexts are associated to each one of them.

Goal creation

The purpose of designing a goal based architecture is to help the robot identifying the intentions of the tutor. A goal is a sequence of contexts that represents a complex behavior. Fulfilling a goal means reproducing the sequence of corresponding primitives according to certain *conditions* set by the actions. Some of these conditions can be inferred from the predefined SN and are learned during the context learning, while the rest are hard-coded in the action associated with the learned context. As an example depicted in Fig. 3, "farness" of an object in "Explore and Reach" action cannot be inferred from the SN since its value changes by each sensor read-outs. Therefore, such a dynamic parameter cannot be represented as a node in the SN. Thus, part of the condition must be hard-coded to always check if the robot has sufficient distance to the object of attention in order to continue execution of the action. The relations between goals, contexts, actions and primitives are illustrated in Fig. 2 and elaborated in the next section.

In order to create a new goal, one has to break down a complex behavior into a set of contexts such that each one represents a behavior primitive. Due to the architectural design, each context maps to one action and each action maps to one behavior primitive. Therefore, complex behaviors are broken down into parts that can be executed by single predefined actions. Each such part is learned as a context. This is done by matching the tele-operation commands during demonstration with hard coded primitives and actions. After finishing the learning process of one context, the tutor starts demonstrating the next context. Environmental conditions help the robot to automatically learn the subsequent context as a sequence of the preceding one.

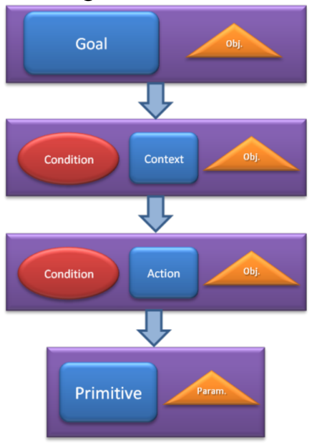

Fig. 2. Relations between Goal, Context, Action, and Primitive

One of the main assumptions is that actions and behavior primitives have a set of pre-defined parameters which applies to objects of attention. Both contexts and actions have objects of attention, which determine on which object to operate. All conditions defined in the actions are checked with the objects of attention.

As long as the conditions are still satisfied, the associated behavior primitive will be executed.

After completion of the high-level learning phase, all learned contexts and their corresponding actions are put together in a sequence, and a new goal object is created. New goals with associated contexts and corresponding actions are stored into a database for retrieval during the reproduction phase.

IV. Experiments

In this section we will present experimental results for learning and reproduction of new contexts. Consider, as an example, the behavior *"Take the Rubble from Human"* as part of an Urban Search and Rescue (USAR) application. The setting is a commercial/residential urban environment damaged by a severe earthquake. The goal for the robot is to assist a human agent cleaning a pile of rubbles covering a victim. The behavior starts with looking for a human agent, getting close to him/her, taking the rubble offered by the human, turning away and reaching the white sign (safe place). Fig. 3 depicts the *"Take the Rubble from Human"* goal, which shows the relations between contexts, their corresponding actions and objects of attention. The rest of the section explains how the robot can learn each context and reproduce the same behavior by perceiving similar environmental conditions.

The behavior is broken down into sub-behaviors such that each context can be associated to one of the pre-defined actions and primitives. It is the responsibility of the tutor to conduct the learning of each context in such a way that it can be associated with an action. The robot starts learning the first context "*Find Human"* as explained in Section III. This is illustrated in the left-most column in Figure 3.

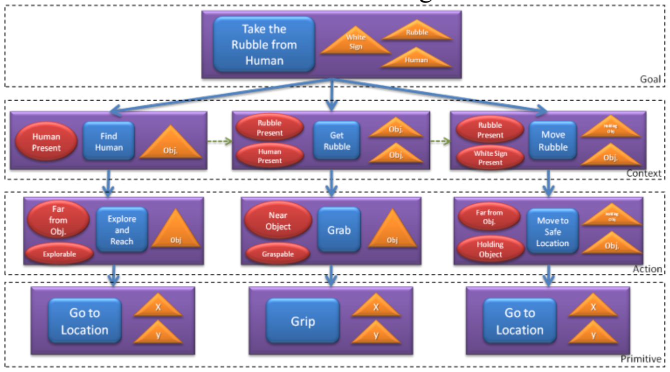

Fig. 3. Structure of *"Take rubble from human"* goal

The "Human Present" condition refers to a node connected to a high-level RFID sensor for detection of humans. Most of the time, an object is detected within a fraction of a second. Therefore, by executing *"Explore and Reach"* action frequently via tele-operation until detecting a human, the robot will establish connections between the *"Find Human"* context node and objects perceived by the RFID tag reader. The tutor must tele-operate the robot until the correct conditions for each context is learned. The pre-defined SN used for learning all contexts is shown in Fig. 4. Some nodes represent concepts and are denoted *category* nodes, the rest represent real objects in the world and are simply denoted *nodes*.

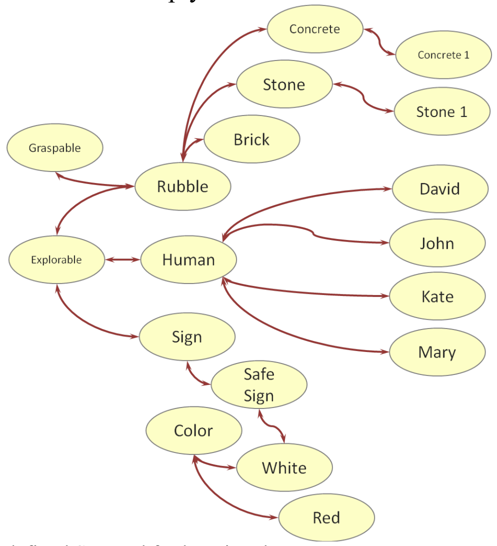

Fig. 4. Predefined SN used for learning the contexts

For learning the first context, *"Find Human"*, the robot recognizes "John" by the RFID tag on his bracelet during the exploration. As a result, the *"John"* node in the SN is activated and will spread the activation to the connecting nodes.

The tutor tele-operates the robot to get close enough to "John" and stops the robot. This signals that learning of the first context is completed. Fig. 5 shows the activation levels of all nodes during the learning of *"Find Human"* context. The behavior has been demonstrated four times with the same person and objects.

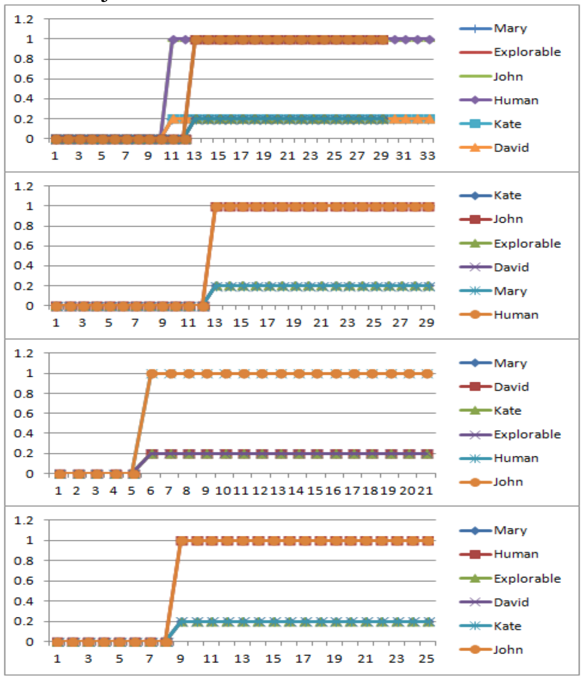

Fig. 5. Node activation levels for learning the *"Find Human"* context perceived in four demonstrations

Based on equations (1) to (10), results are calculated and shown in Table I.

TABLE I
"FIND HUMAN" CONTEXT LEARNING VALUES

Node	BG	WG	T	BDF	WDF	TDF	Cal. F
John	0.403	23.25	23.65	3	104	107	0.6017
Mary	0.016	0.93	0.94	3	104	107	0.6018
Kate	0.016	0.93	0.94	3	104	107	0.6018
David	0.016	0.93	0.94	3	104	107	0.6018
Human	0.403	23.25	23.65	3	104	107	0.6017
Explorable	0.016	0.93	0.94	3	104	107	0.6018

For all nodes, the calculated F-ratio is less than the tabulated value for the F-distribution at significance level $p=0.05$ (2.688), which means that node activations from different demonstrations are from the same distribution, and all nodes should remain connected to the context node.

The weight values for connecting nodes are calculated with equations (11) and (12) and are shown in Table II.

TABLE II
"FIND HUMAN" WEIGHT VALUES

Node	Weight
John	0.403
Mary	0.016
Kate	0.016
David	0.016
Human	0.403
Explorable	0.016

The final relations for the *"Find Human"* context are shown in Fig. 6. The solid links are semantic relations that come from the pre-defined SN and the dashed links are learned during the demonstration.

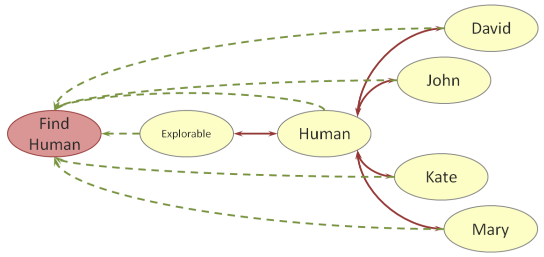

Fig. 6. "Find Human" context

Also features, like *"Explorable"* or *"Graspable",* are represented as nodes in the SN. This facilitates reasoning and allows the system to check if an object of attention can satisfy conditions of the actions.

The learning process of the second context, *"Get Rubble"*, will start after robot reaches "John". "John" picks up a piece of stone (Stone1) and offers it to the robot, while the tutor grabs the piece with the robot arm through tele-operation. While learning this context, the *"Grab"* action command and its linked primitive are executed by the tutor so the robot can recognize the action and associate it with the context.

Through spreading activation, the concepts *"Stone"* and *"John"* will be generalized to *"Rubble"* and *"Human"* respectively. Therefore, the final SN does not only contain the objects perceived, but also similar objects.

For the *"Get Rubble"* context, the tutor demonstrated the behavior four times with the same objects and person. The learning values are listed in Table III.

TABLE III
"GET RUBBLE" CONTEXT LEARNING VALUES

Node	BG	WG	T	BDF	WDF	TDF	Cal. F
Concrete	0.004	0.264	0.268	3	82	85	0.429
Stone	0.323	20.60	20.93	3	82	85	0.429
Rubble	0.104	6.617	6.720	3	82	85	0.429
Brick	0.004	0.264	0.268	3	82	85	0.429
Stone1	0.323	20.60	20.93	3	82	85	0.429
Human	0.051	6.487	6.538	3	82	85	0.215
David	0.002	0.221	0.223	3	82	85	0.271
John	0.054	5.526	5.581	3	82	85	0.271
Mary	0.002	0.221	0.223	3	82	85	0.271
Kate	0.002	0.221	0.223	3	82	85	0.271
Graspable	0.004	0.264	0.268	3	82	85	0.429
Explorable	0.004	0.645	0.650	3	82	85	0.201

The tabulated value F (*p*=0.05) is 2.715, which means that all node activations are from the same distribution and as for the previous context, all nodes should remain connected. The weight values for remaining nodes are shown in Table IV.

TABLE IV
"GET RUBBLE" WEIGHT VALUES

Node	Weight
Concrete	0.0149
Stone	0.1315
Rubble	0.0745
Brick	0.0149
Stone1	0.1315
Human	0.2241
David	0.0421
John	0.2104
Mary	0.0421
Kate	0.0421
Graspable	0.0149
Explorable	0.057

As illustrated in Fig. 7, the *"Get Rubble"* context gets connected to two categories: *Human* and *Rubble*. In the action layer shown in Fig. 3, the *"Grab"* action requires an object that is close and graspable. Thus, the only category that meets this requirement is the *"Rubble"*. Therefore, all the conditions set in the *"Grab"* action are applied to objects in the *"Rubble"* category.

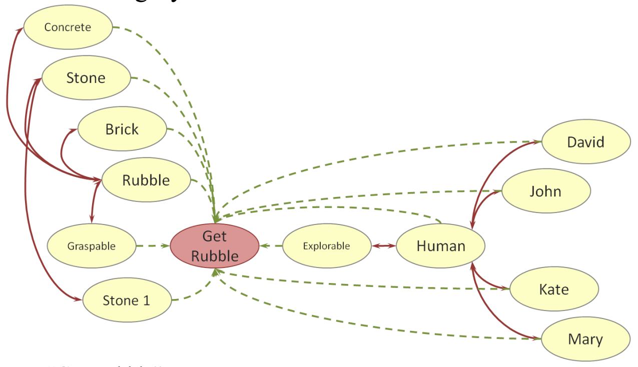

Fig. 7. "Get Rubble" context

The third and the last context to be learned is *"Move Rubble"* which starts when the robot holds the stone. The tutor tele-operates the robot to turn away from "John" and searches for the "Safe Sign". After reaching the designated location, demonstration of the whole behavior is completed.

The same computation is done for the *"Move Rubble"* context after four demonstrations. The computed values are listed in Table V.

TABLE V
"MOVE RUBBLE" CONTEXT LEARNING VALUES

Node	BG	WG	T	BDF	WDF	TDF	Cal. F
Concrete	0.003	0.878	0.882	3	88	91	0.105
Stone	0.042	17.75	17.79	3	88	91	0.07
Rubble	0.005	6.661	6.667	3	88	91	0.026
Brick	0.003	0.878	0.882	3	88	91	0.105
Stone1	0.069	14.4	14.47	3	88	91	0.141
Sign	0.305	23.83	24.13	3	88	91	0.376
Safe Sign	0.069	14.4	14.47	3	88	91	0.141
Color	0.069	14.4	14.47	3	88	91	0.141
White	0.069	14.4	14.47	3	88	91	0.141
Red	0.141	5.565	5.706	3	88	91	0.743
Graspable	0.003	0.878	0.882	3	88	91	0.105
Explorable	0.065	7.166	7.232	3	88	91	0.267

The tabulated value F (*p*=0.05) is 2.708, which is larger than all calculated F. Therefore, all nodes remain connected also for this context. The weight values for remaining nodes are shown in Table VI.

TABLE VI
"MOVE RUBBLE" WEIGHT VALUES

Node	Weight
Concrete	0.0299
Stone	0.1652
Rubble	0.0968
Brick	0.0299
Stone1	0.1503
Sign	0.1091
Safe Sign	0.1056
Color	0.0853
White	0.0853
Red	0.0426
Graspable	0.0299
Explorable	0.0696

What the robot has learned as *"Move Rubble"* is illustrated in Fig. 8.

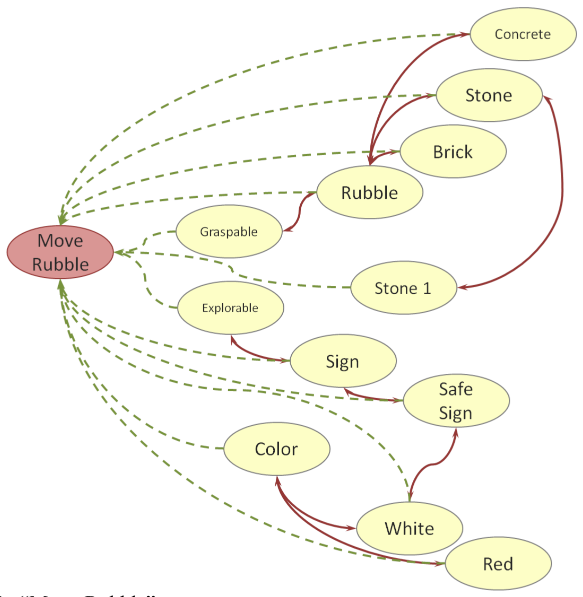

Fig. 8. “Move Rubble” context

Finally, all learned contexts are grouped together automatically to form a new goal that represents the demonstrated behavior (Fig. 3). The sequencing is done automatically by the learning routine that sets the start of the subsequent context to be the end of the preceding one.

V. Goal Inference

During the reproduction phase, the robot pursues a goal that is implicitly or explicitly determined by the user. In both cases, the robot attempts to infer what goals to pursue and activate related contexts in sequence in order to reproduce the learned behavior.

A. *Implicit goal determination*

Implicit goal determination is a bottom-up approach. Perceived objects or concepts activate contexts which in turn activate connected goals. The highest activated goal is selected. The motivation system plays a key role in implicit goal inference by putting the robot into different tracks by stimulating it with cognitive activities such as priming and response facilitation.

Continuing with the USAR scenario, suppose we prime the robot by showing a piece of concrete. As described earlier, concrete was not used directly in teaching but due to the generalization mechanism, contexts with connections to concrete or rubble in general will be activated. Goals connected to these contexts will be determined and listed. The goal connected to the highest activated context will be selected and the actions associated with the first context will be executed. Fig. 9 illustrates the goals and the effect of priming on selecting which goal to execute. In this example, the goal *“Take the Rubble from Human”* is selected.

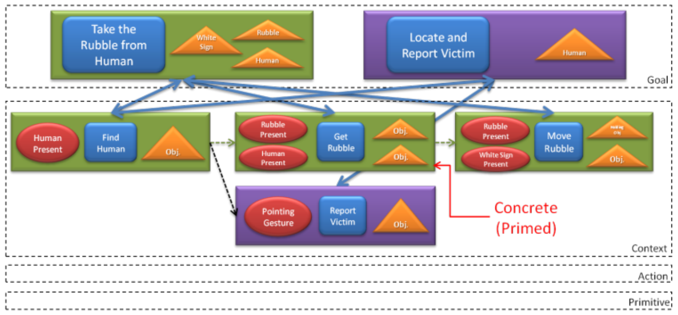

Fig. 9. Implicit goal determination; Priming causes the green goal and contexts to be selected for execution.

The reason is the priming effect which activates both *“Get Rubble”* and *“Move Rubble”* contexts. The activation levels of both contexts totally depend on environmental conditions and perception, but both belong to the *“Take the Rubble from Human”* goal. After determining the goal, the robot begins strolling around and perceiving the environment to fulfill the “Human Present” condition for the first context, *“Find Human”*. Then, it starts executing the action assigned to the context and its corresponding primitive until the action conditions (“Near Object” or “Explorable”) are no longer satisfied. This means that the robot is able to find the human. Now, the second part of the sequence is selected and executed. The robot strolls around again until objects of attention required by the second context are perceived. According to Fig. 9, there are two possible choices as a second context: *“Get Rubble”* and *“Report Victim”*. The latter does not belong to the *“Take the Rubble from Human”* goal. As a result, *“Get Rubble”* and its associated action will be executed until the conditions (“Near Object” or “Graspable”) are no longer satisfied. At this stage, the robot will start the last part of the sequence and finally stop when it reaches the “white safe sign”. In Fig. 9, the robot’s choices are illustrated by green boxes and transitions between the contexts are shown with dashed green arrows. The actions and primitives are shown in Fig. 3.

B. *Explicit goal determination*

Beside implicit goal determination, the user may explicitly specify a goal for the robot. The robot will then select only the contexts that fulfill the specified goal. For instance, if a user specifies *“Take the Rubble from Human”* as goal, the robot will only check for relevant objects and select contexts that are part of the specified goal. Thus, the robot will work top-down to identify the first context of the goal and as a result look for objects of attention defined by the *“Find Human”* context. Depending on the current state of the robot and environment, it may skip executing the first context if it has already reached a human. Thus, it checks for the conditions defined by the *“Get Rubble”* context. The process of context activation and action reproduction continues until the whole sequence is completed. Fig. 10 illustrates the mechanism for explicit goal determination. The green boxes and numbers show how the robot manages the sequential execution of contexts to achieve the selected goal.

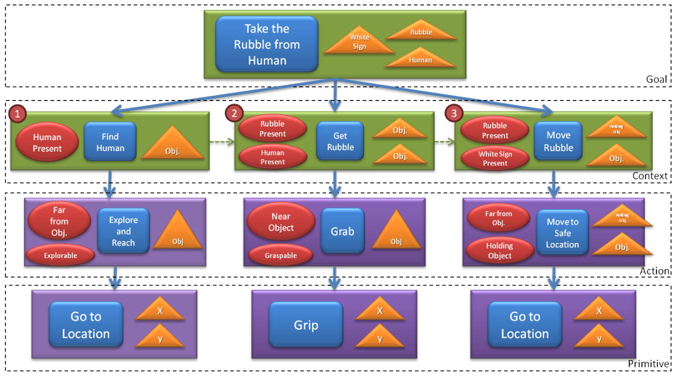

Fig. 10. Explicit goal determination

One of the main strengths of the presented design is the automatic execution of contexts in the right order. This is made possible by the learned conditions that guide the robot to do the right thing at the right time.

VI. Conclusion

In this paper we outlined an architecture for Learning from Demonstration. Considering strengths and weaknesses of other architectures, we proposed a new design for learning high-level representation of the behaviors and associating them with behavior primitives. The modules of the architecture were elaborated and mechanisms for information flow discussed. The Multiple Demonstrations context learning technique was introduced and a mechanism to detect irrelevant nodes was elaborated.

In this research we headed for goal based imitation learning, and by introducing goal creation and inference mechanisms, the robot was able to recognize the tutor's intentions. With the help of the motivation system, the robot can reproduce learned behaviors and pursue specified goals. Finally, the procedure for setting explicit or implicit goals for the robot under the USAR application scenario was discussed.

Acknowledgment

This work was financed by the EU funded Initial Training Network (ITN) in the Marie-Curie People Programme (FP7): INTRO (INTeractive RObotics research network), grant agreement no.: 238486.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, 3(6): pp. 233-242, 1999.

[2] A. Billard, S. Calinon, R. Dillmann and S. Schaal, "Robot programming by demonstration," *Handbook of Robotics*, B. Siciliano and O. Khatib ed., Springer, 2008.

[3] H. Tan, "Implementation of a Framework for Imitation Learning on a Humanoid Robot Using a Cognitive Architecture," *The Future of Humanoid Robots - Research and Applications*, R. Zaier ed., 2012.

[4] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering. "Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning," *Robotics and Autonomous Systems*, 54(5): pp. 353-360, 2006.

[5] C. Chao, M. Cakmak and A. L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," *In proceedings of IEEE International Conference on Development and Learning (ICDL)*, vol. 2, pp. 1-6, 2011.

[6] B. Fonooni, T. Hellström and L. E. Janlert, "Learning high-level behaviors from demonstration through Semantic Networks," *In proceedings of 4th International Conference on Agents and Artificial Intelligence (ICAART)*, pp. 419-426, 2012.

[7] A. Billard and M. J. Mataric, "Learning human arm movements by imitation: evaluation of a biologically inspired connectionist architecture," *Robotics & Autonomous systems*, 37(2-3): pp. 145-160, 2001.

[8] S. Kopp and O. Graeser, "Imitation Learning and Response Facilitation in Embodied Agents," *In Proceedings of the 6th international conference on Intelligent Virtual Agents*, pp. 28-41, 2006.

[9] R. W. Byrne and A. E. Russon, "Learning by imitation: A hierarchical approach," *The Behavioural and Brain Sciences*, Vol. 21, pp. 667-721, 1998.

[10] G. Rizzolatti and L. Craighero, "The Mirror-Neuron System," *Annual reviews of neuroscience*, Vol. 27, pp. 169-192, 2004.

[11] F. Crestani, "Application of Spreading Activation Techniques in Information Retrieval," *Journal of Artificial Intelligence Review*, Vol. 11, pp. 453-482, 1997.

[12] M. L. Mugnier, "On Generalization / Specialization for Conceptual Graphs," *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 7, Issue 3, pp. 325-344, 1995.

[13] M. J. Mataric, "Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics," *Imitation in Animals and Artifacts*, C. Nehaniv and K. Dautenhahn ed., The MIT Press, 2001.

[14] D. C. Howell, "Statistical methods for psychology," *Pacific Grove*, 7th ed., CA, Duxbury, 2010.

[15] Q. Y. Huang, J. S. Su, Y. Z. Zeng, Y. J. Wang, "Spreading activation model for connectivity based clustering," *Advances in Information Systems*, Vol. 4243/2006, pp. 398-407, 2006.

[16] C. Breazeal, M. Berlin, A. Brooks, J. Gray and A. L. Thomaz., "Using perspective taking to learn from ambiguous demonstrations," *Robotics and Autonomous Systems*, 54(5): pp. 385-393, 2006.

[17] E. Oztop, M. Kawato and M. A. Arbib, "Mirror neurons and imitation: A computationally guided review," *Neural Networks*, 19(3): pp. 254-21, 2006.

[18] E. Sauser and A. Billard, "Parallel and distributed neural models of the ideomotor principle: An investigation of imitative cortical pathways," *Neural Networks*, 19(3):285-298, 2006.

[19] Y. Demiris and G. Hayes, "Imitation as a dual route process featuring predictive and learning components: a biologically plausible computational model," *Imitation in Animals and Artifacts*, C. Nehaniv and K. Dautenhahn ed., pp. 327-361, MIT Press, 2002.

[20] C. L. Nehaniv, "Nine billion correspondence problems and some methods for solving them," *In Proceedings of the International Symposium on Imitation in Animals and Artifacts (AISB)*, pp. 93-95, 2003.

[21] D. Sternad and S. Schaal, "Segmentation of end-point trajectories does not imply segmented control," *Experimental Brain Research*, 124/1: pp. 118-136, 1999.

[22] J. Neely, "Semantic priming effects in visual word recognition: A selective review of current findings and theories," *Basic processing in reading: visual word recognition. Hillsdale*, D. Besner and G. Humphreys ed., NJ: Lawrence Erlbaum Association, Inc., pp. 264-336, 1991.

[23] E. A. Billing and T. Hellström, "A formalism for learning from demonstration," *Paladyn Journal of Behavioral Robotics*, 1 (1), pp. 1-13, 2010.

[24] E. A. Billing and T. Hellström, "Behavior recognition for segmentation of demonstrated tasks," *IEEE SMC International Conference on Distributed Human-Machine Systems*, pp. 228-234, 2008.

[25] E. A. Billing, T. Hellström and L. E. Janlert, "Behavior recognition for learning from demonstration," *International Conference on Robotics and Automation (ICRA)*, pp. 866-872, 2010.