Action Rule Induction from Cause-Effect Pairs Learned through Robot-Teacher Interaction

Agostini A., Celaya E., Torras C. and Wörgötter F.

Abstract— **In this work we propose a decision-making system that efficiently learns behaviors in the form of rules using natural human instructions about cause-effect relations in currently observed situations, avoiding complicated instructions and explanations of long-run action sequences and complete world dynamics. The learned rules are represented in a way suitable to both reactive and deliberative approaches, which are thus smoothly integrated. Simple and repetitive tasks are resolved reactively, while complex tasks would be faced in a more deliberative manner using a planner module. Human interaction is only required if the system fails to obtain the expected results when applying a rule, or fails to resolve the task with the knowledge acquired so far.**

I. Introduction

In this work we are facing the problem of decision making for a multitask robot embedded in a human environment that should rapidly learn to perform tasks by interacting with humans, in an on-line way, and without any previous knowledge of the world dynamics or the tasks to be performed.

From a very general point of view, we must consider two alternative approaches to the goal of building an intelligent agent: the deliberative and the reactive approaches. The deliberative approach began with the very birth of AI, and it is based on the principle of rationality [1], which states that "If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action.". The proponents of the knowledge-based systems using the principle of rationality soon realized that there are a number of important shortcomings with this approach, ranging from the frame problem [2], the difficulty of building a large enough database of knowledge providing the grounds for common sense, and the theorems stating the complexity of planning for even some of the simplest kinds of logical problems.

Later, also the symbol grounding and related problems [3] entered the scene. As a response to this, the proponents of the new AI [4] advocated for the reactive approach, in which the knowledge level was completely absent. In this approach actions are not driven by the rationality principle, but triggered by the current situation, and not guided by any specific purpose, but simply as a set of instincts carefully organized to accomplish a specific task.

While reactive approaches have proved to be valid for many low-level tasks, we think that the kind of intelligent behavior we expect from a service robot, like a kitchen assistant, cannot be the result of purely reactive processes. We want the robot to promptly accomplish the task required by the user, and this means that its actions must be goal-driven, and not just situation-driven. We expect the robot to be able to produce new behavior in response to a new goal using its knowledge of the situation and the effects of its actions, but it is clear that a reactive system will only be able to act according to already acquired behaviors.

A number of hybrid approaches have been proposed along these lines. Some of them propose a decision-making system that permits fast agent responses to new situations using reactive layers while the deliberative layers generate behaviors used later by the reactive modules [5]. Others let the low-level action control to be driven by reactive behaviors, which are selected or modulated by a higher deliberative layer [6], [7]. Finally, some works focus mainly on the generation of behaviors such as macro-actions [8], primitive behaviors [9], or activation rules [10], which store sequences of actions frequently used or difficult to calculate, to use them later as macro planning operators in a deliberative system.

In any of the previous cases a large amount of computation is usually required due to the need of exploring different acting behaviors to select one suitable for the task. The problem turns to be more complicated if the robot has no previous knowledge of the world dynamics and should perform learning while predicting what would occur with different behaviors. Incomplete knowledge has been tackled using techniques like incomplete planning [11], learning planning operators [12], [13], [14] or policy learning [15], but the drawback of computational complexity derived of the application of AI techniques is still not surmounted.

The aim of this work is to develop an integrated system in which reactive and deliberative components are both present, though not strictly separated, but smoothly combined, and where the world dynamics and behaviors are rapidly learned from scratch through a natural human-robot interaction.

This work is funded by the EU PACO-PLUS project FP6-2004-IST-4-27657.

A. Agostini is with the Institut de Robòtica I Informàtica Industrial, CSIC-UPC, Barcelona, 08028, Spain (corresponding author) (phone: +34-93-401-5786; fax: +34-93-401-5750; e-mail: agostini@ iri.upc.edu).

E. Celaya, is with the Institut de Robòtica I Informàtica Industrial, CSIC-UPC, Barcelona, 08028, Spain (e-mail: celaya@iri.upc.edu).

C. Torras is with the Institut de Robòtica I Informàtica Industrial, CSIC-UPC, Barcelona, 08028, Spain (e-mail: torras@iri.upc.edu).

F. Wörgötter is with the Bernstein Center for Computational Neuroscience, Göttingen, D37073, Germany (e-mail: worgott@bccn-goettingen.de).

As we want the agent to learn only the dynamics of the world relevant for its purpose, the world exploration is guided by a teacher. It is very simple for humans to know which action to perform in a situation given a plain task, like a kitchen task, but it could be much more complicated to explain a priori all the sequences of actions that should take place in all the possible situations. It might also be difficult for a human to detail all the conditions that should be taken into account to afford a desired cause-effect for all the possible situations. In this work we take benefit of the human capabilities of explaining cause-effect relations in currently observed situations to efficiently generate knowledge for decision making in a multitask robot. The idea is based on Piaget's theory of cognitive development which claims that children gradually acquire knowledge of cause-effect relations by repeatedly executing processes and sequencing actions to reach goals.

This work is organized as follows. Section II explains the outline and main elements of the method proposed. Section III presents a demo application and clarifies some concepts explained in Section II. In Section IV the algorithm is delineated in pseudo-code. A brief discussion of the ideas and concepts of this work in the context of the European project PACO+ [16] is developed in Section V. Finally, section VI delineates some conclusions and future works.

II. Outline of the method

In this work a decision making system is proposed where the action behaviors are generated using simple cause-effect relations learned with the help of a teacher. The learned behaviors are used either reactively or deliberatively depending on the complexity of the task requested.

We will define a behavior (or rule) as a set of preconditions, a sequence of actions, and the final expected outcome. The preconditions are a set of necessary conditions or perceptions that must be observed before the rule can be applied, and the expected outcome is a series of effects that will be obtained after the execution of the rule. The action sequence may consist of a single elementary action in the simplest rules (the cause-effect relation for that action) or a list of actions, each one expressed in turn as a cause-effect.

A general overview of the proposed method is the following. Given a goal, the agent tries to apply any of the existing rules in a reactive way to reach it from the current situation without any deliberation. If more than one rule is retrieved, the one with fewer actions in its sequence is applied. If a reactive behavior is not possible, then the agent tries to generate a plan using the existing rules as planning operators.

If both the reactive and deliberative modules fail to return a behavior, as a consequence of an incomplete knowledge, the agent asks the teacher about which action or actions to perform. The agent executes every instructed action and generates a first approximation of the involved cause-effects by observing the changes in the environment. Then the agent generates a rule with the sequence of the generated cause-effects.

On the contrary, in the case that the agent is able to find a behavior with the reactive or deliberative module, then it executes and evaluates it at the level of each cause-effect in the related sequence. If any of the outcomes obtained is different from the one expected, the agent will ask the teacher for explanations about which conditions prevented the correct outcome of the cause-effect to occur. With the teacher explanation the agent automatically corrects the cause-effect structure as well as all the rules that apply this cause-effect in their sequences performing a large updating of the knowledge base with a little teacher interaction.

A. Notation

We assume that the agent has a set of N sensors that measure some features of the environment. The value of sensor i is called an observation o_i. A world state SO is formed by the set of observations o_i , $SO=\{o_1, o_2, \dots , o_N\}$.

Each of these sensors is internally represented by the agent as a detector d_i that could take different discrete values d_{ij}, called conditions, depending on the sensed value o_i. An internal agent state S is constituted by a set of conditions d_{ij}, $S=\{d_{1j}, d_{2k}, \dots, d_{Nl}\}$.

At every moment the agent is able to perform any of the k actions from the set $A=\{a_1, a_2, \dots, a_k\}$.

The function that maps the sensor observations to conditions is called the *perception function* (*PF*). As we will explain later the *PF* could be updated while the learning process is running, permitting the management of the uncertainties, inherent to real environments.

The most elementary rule consists of a cause-effect relation and reflects how a change is obtained using a single action and what preconditions are necessary to afford that change. We formally represent a cause-effect cec_i using a tuple that consists in a subset P_i of state conditions called the preconditions of the cec_i, an action a_i from the set of actions A, and a subset O_i of state conditions denoted as the expected outcome of the cec_i.

$$cec_i = <P_i=\{d_{gj},\dots,d_{ml}\}, a_i , O_i=\{d_{kl},\dots,d_{pq}\}> \qquad (1)$$

In the same way, a rule R_j is described using a tuple that consists of a subset P_j of state conditions called the preconditions of the rule R_j, a sequence of *cec's* $CECS=(cec_k,\ cec_i, \dots,\ cec_m)$, and a subset O_j of state conditions denoted as the expected outcome of the rule.

$$R_j = <P_j=\{d_{ih},\dots,d_{ml}\}, CECS , O_j=\{d_{kl},\dots,d_{pq}\}> \qquad (2)$$

In our approach, the expected outcome serves two purposes: it will be used by a goal-achieving deliberative system for planning, and by a learning system to improve rule descriptions. Every time the expected outcome is different from the observed we will say that the robot gets a *surprise*.

B. Learning Rules

When the knowledge base of the system doesn't permit to find a rule, or a sequence of rules, to be applied in an experienced situation, the teacher instructs the robot about which action or sequence of actions to execute. Then, the robot executes every instructed action generating a first approximation of the involved cause-effects, and afterwards builds a rule using the sequence of the generated *cec's*.

1) Generating cecs

The robot generates a first approximation of the cause-effect observing the conditions that change in the states before and after the execution of the instructed action *a*. If we call the state before the action execution S^{prior} and the state after the action execution S^{post} the new cec_{new} is:

$$cec_{new}=< P_{new}, a, O_{new}> \quad (3)$$

where,

$$P_{new} = \{ d_{ij} \in S^{prior} \mid d_{ij} \notin S^{post} \} \quad (4)$$

$$O_{new} = \{ d_{kl} \in S^{post} \mid d_{kl} \notin S^{prior} \} \quad (5)$$

The preconditions of the cec_{new} so formed could be incomplete in the sense that there could be conditions that do not change before and after action execution, but are also necessary to produce the changes observed (for example, the density of an object that prevents its deformation, the friction of a surface that prevents an object displacement, etc.). In these cases the teacher would explain which conditions are missing to obtain the expected outcome.

2) Generating Rules

The generated *cec's* are used to create rules that will contain the *cec's* sequence. The preconditions P_{Rnew} of the rule R_{new} formed should ensure the occurrence of the *cec's* preconditions in the proper order. For those detectors that take only one condition value during the sequence, it is straightforward that this value should also appear in the rule preconditions. If there is more than one condition for a particular detector, the one closer to the origin of the sequence should occur first, and this one should be in the rule precondition. Therefore, the rule preconditions can be obtained directly by back-propagating with replacement all the preconditions of the *cec's* departing from the last *cec* to the first one. In contrast, to deduce the final outcome of the *cec's* sequence, and hence of the rule O_{Rnew}, we should take into account the last changes produced in each detector. Therefore, if there is more than one condition for a detector, the one that should be considered for O_{Rnew} is the farthest from the origin. We can obtain all the conditions of the rule outcome again by back-propagating the conditions of the *cec's* outcomes, but now without replacement departing from the last *cec* to the first one. The process of rule generation is illustrated in Section III.

There are two remarkable aspects. The first one is that, assuming all the proper preconditions are considered, the rules would produce the expected outcome in all the situations where the respective preconditions are present, despite some of these situations not having ever been experienced before. Therefore, rules perform generalization over all the situations where the corresponding sequence would take place. The second notable characteristic is that, for a given sequence CECS of *cecs*, as many rules as sub-sequences in CECS could be generated, using the initial and final *cec* of the sub-sequences as the initial and final points for the back-propagation procedure. The subset of rules actually generated depends on the criterion adopted. For instance, the robot could be required to only learn how to reach the goals specified by the teacher, leading to the generation of only the subset of rules consisting of every sub-sequence from an intermediate situation to the goal.

C. Rule Correction

During the execution of a rule the robot can get a surprise if one of the involved *cecs* results in an unexpected outcome. Then, the teacher "explains" which preconditions prevented the expected outcome to occur. The reason of the surprise could be produced by either a missing condition in the precondition part or by a wrongly interpreted condition due to a problem in the perception function *PF*. In both cases the teacher tells the robot which conditions are responsible for the failure, specifying the detectors and the corresponding values. The explanation given is used to update the *PF* and to correct the *cec*. The explanation could be indeed incomplete, not specifying all the conditions that would prevent the expected outcome to occur, but only those that the teacher is capable of identifying at that moment as the ones responsible for the surprise. This is accepted as far as the teacher is able to realize in future observations the other conditions that are responsible for the failure.

After the *cec* correction, the rules correction is simple and straightforward. It is performed by updating all the rules that contain the corrected *cec* in their sequences just by back-propagating the explained conditions as explained in the rule generation section.

D. On Learning to Perceive

We want to briefly remark the underlying idea about how the perception function *PF* could be updated using the teacher explanations. The idea is expressed in the scope of simple applications (like the one presented in Section III) where the perceptions of the robot could be derived by the sensor observations using a probabilistic approach.

If we assume that the sensor observations o_i are continuous variables with uncertainties and non-stationarities, the way to correctly map a value o_i to a condition d_{ij} is difficult to establish a priori. It is possible to face this matter through a probabilistic approach that for each condition d_{ij} permits to infer how probable it is that a sensed value o_i is interpreted as d_{ij}. Then, for a particular observed value o_i, the condition d_{ij} perceived is the one with highest probability in o_i. The estimated statistic values related to a condition d_{ij} could be updated using the teacher explanations on this condition using the corresponding observed value o_i. This updating permits the teacher also to explain the robot how to interpret the world.

III. Demo Application

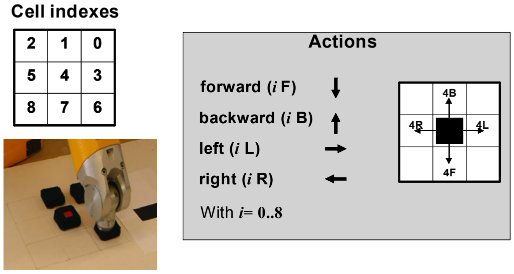

Fig. 1. Demo application elements.

Figure 1 shows a schema of a simple real world application implemented in a Staubli arm that permits to visualize the important aspects of the method performance. The application consists in an environment with 9 cells configured in a 3 by 3 grid world. Each cell could contain a black box or be empty. The amount of boxes that can be placed in the grid ranges from 1 to 8. Among all the boxes there is one target box marked with a red label. The task consists in placing the target box into a goal cell without taking any box outside of the grid. To this purpose the arm can move, when it is possible, any box form its current cell to one of the contiguous cells in straight line (diagonal movements are not allowed). Movements that take any box out of the grid cannot be performed.

A cell is considered as a detector in the state representation. The state is represented graphically in the examples, where a black cell represents a box in that cell, a white represents an empty cell and a black with a red mark represents the cell containing the target box. Dashes cells mean "don't care" if there is either a box or an empty space. Figure 1 explains the possible actions and how the cells are indexed to reference detectors and actions.

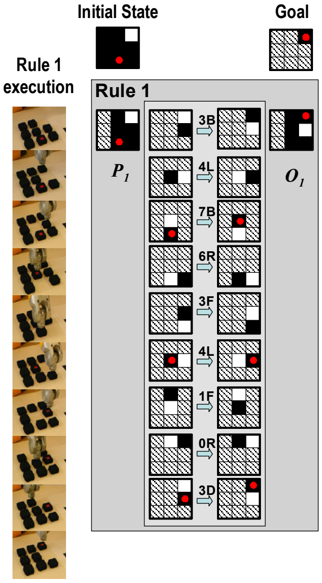

Fig. 2. The largest rule generated after the instructions *iseq1* to take the target box from cell 7 to cell 0.

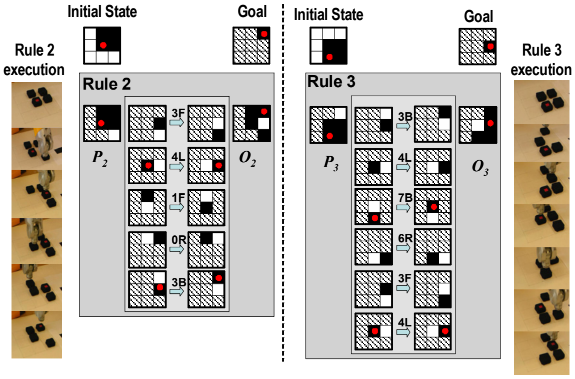

Fig. 3. Execution of two rules generated from *iseq1* under different goals requirements.

Before presenting some results we would like to mention that the rule generation criterion adopted for this example is to generate as many rules as sub-sequences there are in the instructed sequences. The robot started the experiments without any previous knowledge. Due to space restrictions the process of instructions is not shown graphically but mentioned during the descriptions of the experiments.

The first instruction received by the robot was to move the target box from cell 7 to cell 0 with the grid full of boxes except cell 0 which was empty. This instructed sequence will be referenced in the following as *iseq1*. Figure 2 shows the largest rule generated from *iseq1* and snapshots of the rule execution given an initial state and goal where the rule was applicable. Figure 3 shows two more rules generated with *iseq1* executed under different requirements of goals and with initial states never experienced before by the robot. The possibility of resolving situations never experienced elucidates the generalization capabilities of the method.

We now instruct the robot to move the target box from cell 5 to cell 4, when cell 4 is occupied and cell 7 is empty.

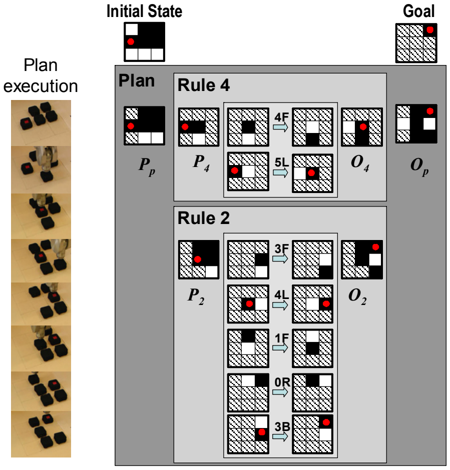

Fig. 4. Plan that linked rule 4 and rule 2 to fulfill a given requirements of initial state and goal.

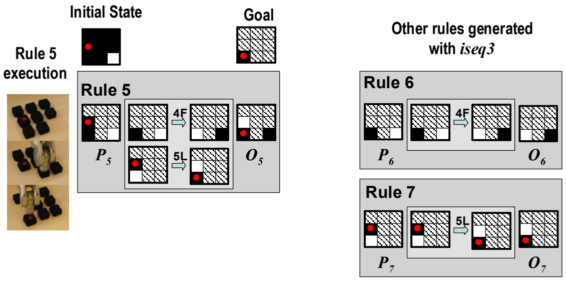

Fig. 5. Rules generated with *iseq3*.

The instructed sequence is denoted as *iseq2*. As a consequence of *iseq2* the robot generated rule 4 (see figure 4). Next, to show how the planner module is activated when no rule is reactively triggered the target cell was placed again in cell 5 but now four more boxes were added in the grid configuring an initial situation shown in figure 4. The robot was then asked to take the target cell from cell 5 to cell 0. For these requirements there was no rule in its database that permitted a reactive behaviour. The planner module was then activated and a plan, that linked rule 4 with rule 2, was found and executed. Figure 4 also suggests how a plan could be transformed into a new rule using the condition propagation.

A. *Surprise and Explanation*

In this section we exemplify how a surprise arises and how the explanations are used to correct the incomplete *cec* and the rules that involve it. First we instructed the robot to move the target box from cell 5 to cell 8 when cells 8 and 7 are occupied and cell 6 is free. This instruction is referred to as *iseq3*. Figure 5 shows the execution of the largest rule generated with *iseq3*, as well as the other two rules generated with the same sequence. Note that for the first action instructed the robot pushed boxes in the cells 8 and 7 but the state representation for cell 7 remained the same before and after the action execution. Hence the generated *cec*, which extracted only the conditions that changed, initially contained a "don't care" in that position.

Afterward, in figure 6, we made the robot to face a problem where the initial state and goal triggered rule 5. The first *cec* execution led to a surprise as the obtained outcome was not included in the expected ones. The teacher then explained that a black in cell 7 should also be considered and the robot corrected the *cec* as well as the involved rules.

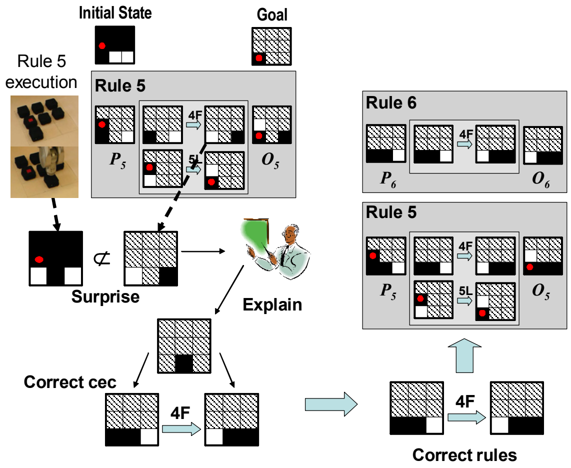

Fig. 6. A schema of the surprise-explain-correct process.

Finally, we made the robot face the same problem that previously resulted in a surprise but then no rule could be applied reactively. Nevertheless, the robot found a plan using one of the rules generated with *iseq1* (rule 8) and the recently corrected rule (rule 5) as illustrated in figure 7. The plan found is not the optimal way to solve the problem because the robot was only able to use the limited knowledge acquired so far. It is important to mention that, in case many plans are found, the robot uses the same criterion as with the rules, i.e., it selects the one with fewer actions.

IV. Sketch of the Algorithm

In this section we present the whole method in pseudo-code. It is important to remark that, in this first approach, we let the teacher control the rule generation by the instruction given. The teacher will instruct a single action when no sequence is convenient to be merged in a rule, and will instruct a sequence of actions for repetitive sequences.

A. *Pseudo-code*

```
INIT system RR={}, LCECS={}, CECS={}
Define GOAL
Sprior=PF(SOprior)
WHILE goal is not reached
  RR: rules that connect Sprior to GOAL
  IF RR is not empty (Reactive)
    Select rule of RR with fewer cecs in CECS
    Execute CECS
  ELSE (RR is empty)
    Try to find a PLAN with the rules.
    IF PLAN is possible (Deliberative)
      Execute CECS
    ELSE (If plan is not possible)
      Teacher instructs actions
      FOR each action instructed,
        Sprior = PF(SOprior)
        Execute action
        Spost = PF(SOpost)
        GENERATE new cec using Sprior and Spost
        APPEND the cec to LCECS
      END FOR
      GENERATE RULES using LCECS
    END ELSE (planning not possible)
  END ELSE (RR is empty)
  Sprior=PF(SOprior)
  Teacher supervises if Sprior is well perceived
  IF Sprior is wrongly perceived
    Teacher explains bad conditions
    UPDATE PF
    Correct Sprior
  END IF
END WHILE (goal is not reached)
```

1) *Execute CECS*

```
FOR each cec in CECS
  Execute action
  Spost=PERCEIVE(SOpost)
    IF not the expected outcome (SURPRISE)
      Teacher explains bad/missing conditions
      Correct cec with teacher explanations
      Correct rules containing the cec
      Update PF
      EXIT FOR
END FOR
```

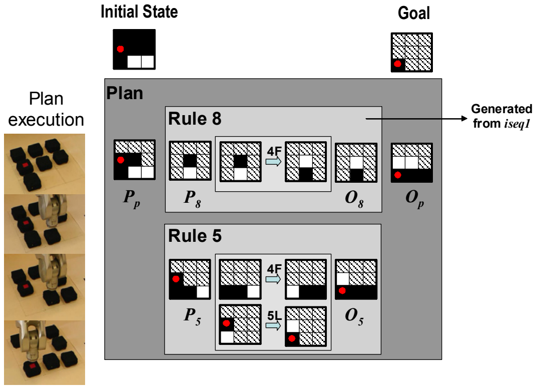

Fig. 7. Plan execution involving one rule generated with *iseq1* and the corrected rule generated with *iseq3*.

V. Discussion in the Context of PACO+

Most of the "learning to act" approaches are based on human learning and cognition capabilities. Despite these approaches present many differences among them, they all establish a direct relation between perceptions of the agent, coded mainly as states, and actions. In contrast to the amount of approaches developed, only few attempts were aimed at creating a common framework that permits to consistently relate the learning to act approaches with the human cognition capabilities for learning and acting. One of these attempts is the concept of object-action complexes (OACs) [17] that has been evaluated and developed by the European PACO+ consortium [16]. Briefly, the OAC concept claims that the world contains undistinguished "things" meaningless for the agent that only become meaningful "objects" through actions and tasks, where the objects are described by the properties relevant for the fulfillment of the final desired outcome through the action.

We believe that the explicit coding of the world conditions and actions through rules and cause-effects presented above is suitable for a first insight in the study and refinement of the OAC concept. One of the reasons is that the elements of these structures could be directly associated with the main elements of the OACs concept formulated so far. Another reason is that they permit a direct association with the human cognition capabilities through the explicit declaration of the abstract meaning of the conditions of the world, and hence a better understanding and a faster evaluation of the results.

VI. Conclusion and Future Extensions

Despite the advantages presented in using simple human instructions for learning to perform tasks, the system should be also able to perform a task without the help of any human in case there is none available. This could be fulfilled by giving the instructions and explanation by other embedded automatic systems. The instruction could be given by an incomplete planner establishing some criterion for rule generation with a measure of the frequency of usage and the amount of computational process needed to generate a given plan. The explanation could be replaced by a constructive learning system where, for instance, a memory-based system could permit to infer which conditions are responsible for the surprise [12], [13], [14]. We believe that the presented method establishes a very suitable platform for future extension to develop a robust decision-making system for a complex robot interacting in a human environment.

References

[1] A. Newell. "The knowledge level". *Artificial Intelligence*, 18(1), 87-127, 1982.

[2] J. McCarthy and P.J. Hayes. "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in *B. Meltzer and D. Michie eds.*, *Machine Intelligence*, Edinburgh: Edinburgh University Press, 1969, pp. 463-502.

[3] S. Harnad. "The Symbol Grounding Problem". *Physica* D 42: 335-346, 1990.

[4] R. Brooks. "Intelligence without representation". *Artificial Intelligence*, 47, pp. 139-159, 1991.

[5] M. Lemaître, G. Verfaillie. "Interaction between reactive and deliberative tasks for on-line decision-making" presented at the 2007 International Conference on Automated Planning and Scheduling, Providence, Rhode Island, USA, 2007.

[6] E. Gat. "On three-layer architectures" in *D. Kortenkamp, R. P. Bonnasso, and R. Murphy, editors*. *Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, pp. 195-210, 1998.

[7] M. J. Schoppers. "Universal Plans for Reactive Robots in Unpredictable Environments", in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI 87)*, Milan, Italy, 1987, pp. 1039-1046.

[8] M. Newton, J. Levine. "Evolving Macro-Actions for Planning", presented at the 2007 International Conference on Automated Planning and Scheduling, Providence, Rhode Island, USA, 2007.

[9] M. Nicolescu, M. Mataric. "A Hierarchical Architecture for Behavior-Based Robots" in *Proc. of the 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, Bolgna, Italy, 2002, pp. 227-233.

[10] D. Rao, Z. Jiang, Y. Jiang. "Learning Activation Rules for Derived Predicates from Plan Examples" presented at the 2007 International Conference on Automated Planning and Scheduling, Providence, Rhode Island, USA, 2007.

[11] S.Yoon, S. Kambhampati . "Towards Model-lite Planning: A Proposal For Learning & Planning with Incomplete Domain Models" presented at the 2007 International Conference on Automated Planning and Scheduling, Providence, Rhode Island, USA, 2007.

[12] X. Wang. "Learning planning operators by observation and practice", in *Proceedings of the Second International Conference on AI Planning Systems*, Chicago, IL, USA, 1994.

[13] T. Oates and P. Cohen. "Learning planning operators with conditional and probabilistic effects", in *Proceedings of the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*, 1996, pp. 86-94.

[14] S. Benson. "Inductive learning of reactive action models", *in Proceedings of the 12th International Conference of Machine Learning*, 1995, pp. 47–54.

[15] R. Sutton and A. Barto. "Reinforcement Learning. An Introduction". *MIT Press*, 1998.

[16] http://www.paco-plus.org

[17] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krüger and F. Wörgötter. "Object Action Complexes as an Interface for Planning and Robot Control", *presented at IEEE RAS Int Conf. Humanoid Robot,* Genova, Italy, 2006.