Towards Practical and Grounded Knowledge Representation Systems for Autonomous Household Robots

Moritz Tenorth, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{tenorth, beetz}@cs.tum.edu

***Abstract*— Mobile household robots need much knowledge about objects, places and actions when performing more and more complex tasks. They must be able to recognize objects, know what they are and how they can be used. This knowledge can often be specified more easily in terms of action-related concepts than by giving declarative descriptions of the appearance of objects. Defining chairs as objects to sit on, for instance, is much more natural than describing how chairs in general look like. Having grounded symbolic models of its actions and related concepts allows the robot to reason about its activities and improve its problem solving performance. In order to use action-related concepts, the robot must be able to find them in its environment. We present a practical approach to robot knowledge representation that combines description logics knowledge bases with data mining and (self-) observation modules. The robot collects experiences while executing actions and uses them to learn models and aspects of action-related concepts grounded in its perception and action system. We demonstrate our approach by learning places that are involved in mobile robot manipulation actions.**

I. Introduction

In order to avoid the need for hardcoding huge parts of autonomous robot systems — both simulated and real ones — we must equip robots with an enormous amount of knowledge. Even accomplishing seemingly simple tasks such as setting the table require the robot to know what cups, glasses, drawers, cupboards, chairs, and many other things are. Robots must be capable of *recognizing* these objects and know how to grasp, carry them etc. Robots must also know which utensils are used for breakfast and how they are to be arranged on the table, where the utensils and their arrangement might be context dependent and change over time.

While for many of these concepts it is difficult and tedious to state what they are and how they look, they can be easily specified in terms of their roles in actions and activities. This is because objects in our household are made for performing, enabling, and supporting certain actions. Chairs are objects to sit on. Cups and glasses differ because they are supposed to be grasped in different ways. Also, people go to to places in order to do something: they go to the cupboard to retrieve cups and sit at the table to eat.

As a consequence, many objects and places of our daily life can be specified both naturally and easily in terms of the roles they play in actions and activities. While the action-related concept definitions are elegant and easy to agree with, they have a big disadvantage: the specifications are not effective, meaning that we can only recognize the concepts *after* the respective actions have been taken.

To achieve both, we propose to equip robots with knowledge representation and reasoning systems that can automatically learn descriptive concept specifications from action-related ones. To this end, we propose to extend description-logics based knowledge representation systems of robots with means to represent, learn, and employ action-related concepts based on the following principles.

- The *knowledge base* includes manipulation actions such as pick, place, transport, different classes of objects relevant for kitchen work, locations, and situations. Using these concepts, a programmer can define new concepts such as the utensils needed for breakfast or the locations where people stand when taking cups from the cupboard.
- Robots are equipped with an *observation system* for their own actions and activities as well as for those of other agents. This observation system transforms observations of actions into abstract learning-task specific experiences that include parameterizations of actions, the robot's beliefs and intentions of the robot, aspects of the action context that might affect the action and their execution, and features of objects manipulated.
- The representation provides mechanisms for *learning probabilistic classifiers* for these concepts that decide whether or not a given entity belongs to the respective concept. The classifiers are to be expressed in a given feature language, such as the set of perceivable features.
- The concepts and their learned descriptions are *first class objects of the knowledge representation languages* which enables their use in other inference tasks or for defining more advanced and derived concepts. The control system can also use the learned models for predicting properties of actions and their effects, for inferring intentions, guessing action parameters, classifying behaviors, etc.

In this paper, we extend GrAM (Grounded Action-related Models, [1]) and apply the approach to mobile manipulation actions performed by an autonomous household robot in simulation and reality. The system is using PROLOG as the basic reasoning engine. The learning, data mining, and reasoning mechanisms needed for grounded knowledge representation are integrated by extending the set of predefined PROLOG predicates. OWL (Web Ontology Language) serves as the

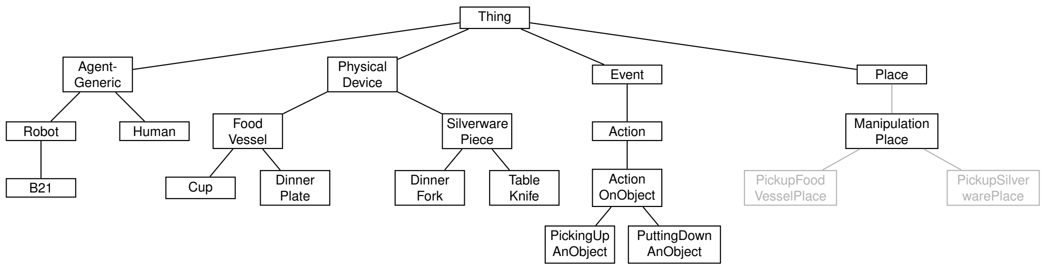

Fig. 1. Simplified structure of the robot's knowledge base.

storage and interchange format for our knowledge bases.

Thus a grounded knowledge base for a kitchen robot can be built by interacting with the robot, by defining new action-related concepts in the robot ontology, by collecting experience during robot operation, and by using the PROLOG tell and ask interface.

The main contributions of this paper are

- the proposition to specify objects, places etc for mobile robots manipulation tasks by their role in actions rather than their appearance,
- a method for autonomously learning grounded models of implicitly specified action-related concepts from collected experiences,
- a system architecture that combines a knowledge base, (self-) observation modules anchored in the robot's perception and action system and data mining methods to learn action-related concepts.

In the remainder of this paper we proceed as follows. In the next section we give an example that demonstrates how our system learns action-related places for the table setting task and how it uses the learned concepts in order to infer the intentions of people. The subsequent sections describe the technical aspects of our system. We start with describing the knowledge base, then present the observation system for actions and finally explain how the models are learned. The paper ends with a review of related work and our conclusions.

II. Usage Scenario

In order to get some intuition of how our knowledge representation system works, we start by describing an interactive knowledge acquisition session in which a programmer identifies concepts that are important for the robot's operation, specifies them in action-related terms, and develops additional characteristics of these concepts through the application of knowledge mining mechanisms to experience data acquired by the robot during its operation. An overview of the system he is using is given in Figure 2.

The programmer starts with the taxonomy depicted in Figure 1 without the concepts in gray which are the ones that will be acquired in this session. In the beginning, the taxonomy includes the concepts of a robot, of physical devices with cups and plates, events with actions, pick-up and place actions as specializations, and places including manipulation places.

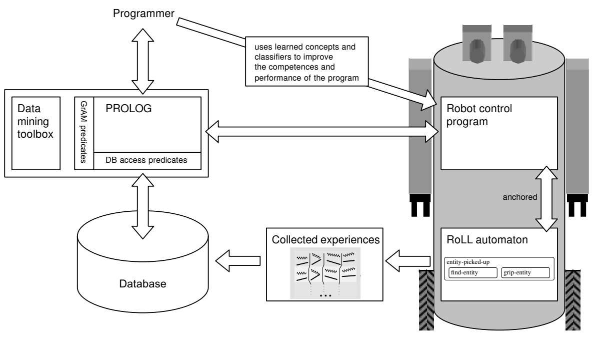

Fig. 2. Structure diagram of our system. Information about the execution of actions is recorded and stored in a database. The reasoning system can access this data and derive action-related concepts.

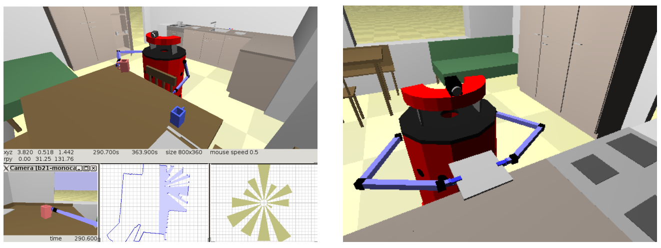

Fig. 3. Kitchen scenario with a simulated B21 robot equipped with a camera, laser, sonar sensors, and two arms.

The robot (shown in Figure 3) has performed kitchen tasks and stored experience data collected over this time into a database that can be accessed using GrAM predicates. Using these experience data, the programmer now wants to refine the knowledge base by identifying the places that are relevant for the robot's performed tasks and represent these places in the knowledge base. The programmer starts by looking at the table setting episodes of the robot by defining the relevant activity episodes:

```
tell tablesettingTask(T) :-
        task(T),
        task-goal(T, table-set-for(M)),
        breakfast(M).
```

In this example, a *tablesettingTask* is defined as a task with the goal *table-set-for(M)* where *M* is a breakfast event.

Because table setting is a mobile manipulation task, he is interested in where the manipulation actions during the breakfast take place. Thus, he queries the positions where pick up and put down actions occur and asserts these places to the knowledge base.

```
tell manipPositions(MPs) :-
       setof(P, manipPosition(P), MPs).

tell manipPosition(P) :-
       (task-goal(Task, pickingUpAnObject);
          task-goal(Task, puttingDownAnObject)),
       subtask (Task, T),
       tablesettingTask(T),
       task-start(Task, Tstart),
       holds(position(Robot, P), Tstart).

ask manipPositions(ManipPosSet).
```

Manipulation places are defined as the positions where the robot stands when starting to pursue a goal *pickingUpAnObject* or *puttingDownAnObject* as a subtask of setting the table. The result of the GRAM query, i.e. the binding of the variable *ManipPosSet*, is visualized in Figure 4.

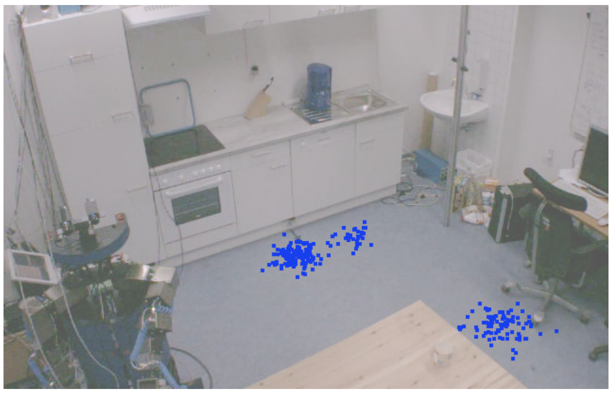

Fig. 4. Raw positions of manipulation actions (dark blue dots), recorded while the robot (bottom left) set the table in our demo kitchen. Due to the noisy measurements, it is hard to find a link between actions and positions.

Now the programmer abstracts the positions into places. In GrAM, places are clusters of positions with respect to the Euklidean distance of the postions as cost function. To support the assertion of places, GrAM provides the built-in predicate *possiblePlaces(PosSet, Places)*.

This predicate applies clustering algorithms to *PosSet* and stores the result of the clustering into the variable *Places*. The predicate backtracks over different algorithms for clustering and different parameterizations of them.

Thus, the programmer states the query *possiblePlaces(ManipPosSet ManipPlaces)* and backtracks until he finds that the proposed *ManipPlaces* abstract *ManipPosSet* adequately, which is depicted in Figure 5. When he obtained a good solution, he can assert it to the knowledge base.

In GrAM, sets of n-dimensional data points can also be abstracted into n-dimensional probability distributions. Probability distributions are first class objects in GrAM which allow for special purpose inferences such as sampling from them. Thus, our programmer represents a place by a probability distribution over positions (see Figure 6) and samples from the distribution for the reinforcement learning of pickup actions of the robot.

Next, the programmer investigates whether the robot performs its manipulation tasks at particular places. Thus, he wants to know if we can predict where the robot will perform a particular action based on the kind of action that is intended

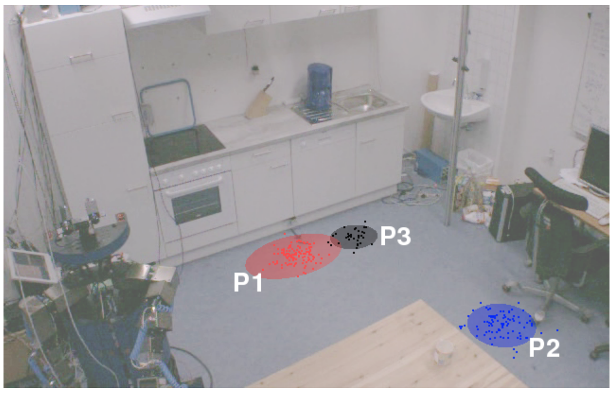

Fig. 5. Clustering positions of manipulation actions creates more natural "places" that can be set in relation to the type of action performed from there.

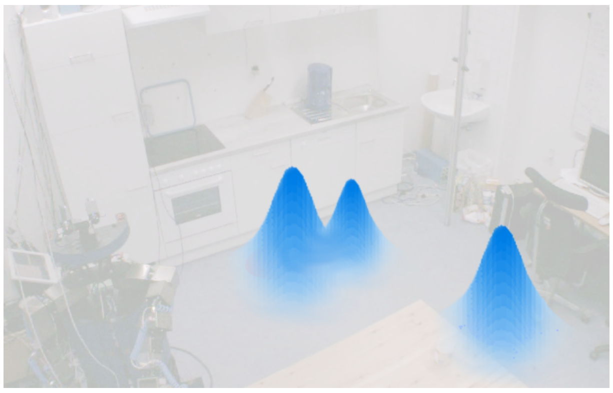

Fig. 6. Probability distribution of the robot position in manipulation actions, estimated based on observed positions of the robot.

and the object the action is performed on. Based on the data collected during the robot's operation, which is shown in Table I, the programmer wants to learn prediction rules for the place where an action is performed.

Agent	Object	Object class	Action	Place
B21	DinnerPlate	FoodVessel	PickingUpAnObject	P1
B21	Cup	FoodVessel	PickingUpAnObject	P1
B21	TableKnife	SilverwarePiece	PickingUpAnObject	P3
B21	DinnerPlate	FoodVessel	PuttingDownAnObject	P2
B21	DinnerFork	SilverwarePiece	PuttingDownAnObject	P2

TABLE I

DATA COLLECTED FROM EXPERIENCES FOR THE "SET THE TABLE FOR BREAKFAST"-TASK.

In order to keep these rules as general as possible, our programmer includes not only the objects themselves, but also the classes they belong to. He already modeled the taxonomy of objects in the knowledge base, so he can include the same categories the robot will later use.

This learning problem can be described by the intensional model named *manipPlaceModel* shown below. It represents the correlation between the stated observable action properties *agent*, *object*, *objectclass* and *action* on the one side and the *place* from which the action was performed on the other. The model is based on all observed places involved in manipulation actions (denoted by *forPlace* and the *pickup* and *putdown* predicates).

```
tell intensionalModel(manipPlaceModel,
     [ forPlace(Ps),
       observable([agent, object, objectclass, action]),
       predictable([place])])
     :- setof(P, (pickup(P, _); putdown(P, _)), Ps).
```

An intensional model is a general description of a model for an action-related concept. When being accessed, an *extensional* model is built by training a classifier that extracts the relations between the observable and predictable properties specified in the *intensional* model.

The extensional model consists of rules that map from a combination of observed values to the respective prediction with a certain probability. This probability is based on the amount of training data that supports the rule and can be obtained from the learned classifier.

An example rule is given below, which states that an action will be performed from place *p1* if its type is *pickingUpAnObject* and if the object is of the class "FoodVessel". The rules that form the extensional model can be asserted to the knowledge base to extend the place hierarchy and define more specific *manipulationPlaces*.

```
tell rule(pickFoodVesselPlace,
     [ body(
        [ withAction(pickingUpAnObject),
          withObjectClass(FoodVessel) ]),
       head(place(p1)),
       withProbability(1.0)])
```

We would like to point out that intensional models are completely independent of the context of the robot. They just describe in general which data can be used to build a model for a concept. Only the extensional model which is learned from the robot's experiences depends on the actual environment.

Using the concept of a pickFoodVesselPlace for the setTheTable activity, the robot can learn the conditional probability over the subsequent manipulation actions given that a person is entering the pickFoodVesselPlace while setting the table. This can, as depicted in Figure 7, be used to infer the context if a person picks up a cup upon entering this place:

Manipulation places in combination with the associated actions and objects can also be used to characterize robot plans. Training a Conditional Random Field (CRF) on a sequence of actions, objects and places allows to analyze the plan structure, common action sequences and their transition probabilities. When interpreting human activities, CRFs can also serve for matching a sequence of actions to the robot's own plans. If the robot has a transformational planning system, it can thereby determine possibly advantageous plan transformations and learn from observing human behavior.

Grounded models of action-related concepts form the basis for a variety of applications that help the robot understand its environment and improve its behavior. Our approach can be used to learn very different kinds of concepts and properties, including possible states of an object (empty or filled cup, open or locked cupboard), special grasping strategies for certain (fragile) objects or classes of similar objects that look very different (sofa, stool, office chair,...).

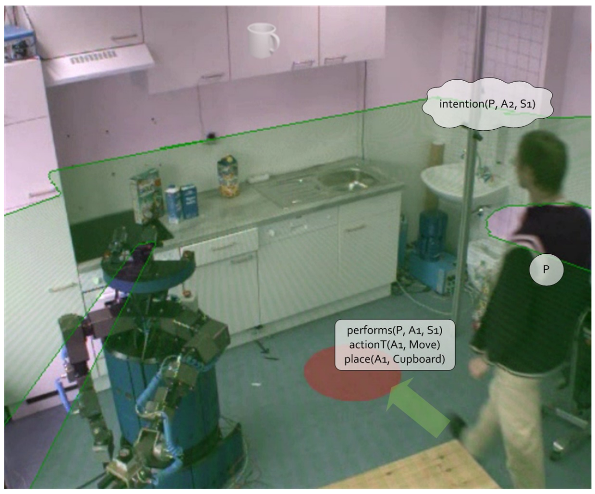

Fig. 7. Recognizing human intentions allows the robot to react, i.e. either to support the human or to keep clear. Action-related concepts like the places where one stands when performing an action can help the robot decide what a human is currently doing.

III. Implementation

The proposed system consists of three main modules, the knowledge base, methods for data collection and the data mining and model construction which are explained in more detail in the following sections.

A. Knowledge Base

The concepts in the knowledge base describe objects, agents, actions and other phenomena from the robot's environment. Since our robot operates in the real world, it needs access to general common-sense knowledge to successfully plan and perform its tasks. Therefore, we derived our knowledge base from the Cyc upper ontology that provides a large number of concepts, categorizations and properties.

However, Cyc was built as a general basis to represent various kinds of knowledge in a common framework, which means that it is both too large and too general for our needs. Information about literature or military operations, for instance, is not needed by a household robot, but detailed models of tasks and objects are crucial. The robot has to know how to grip an object, what an object can be used for, what not to do with fragile or dangerous objects et cetera. Therefore, we decided to use the Cyc ontology as the upper layer, but extend it to better represent object manipulation actions.

Our knowledge representation is based on PROLOG as a powerful, expressive reasoning system. Extensions have been made to integrate the data mining and learning modules which are automatically triggered if a respective concept is accessed.

B. Observation System and Data Acquisition

In order to build grounded knowledge bases the robots need mechanisms for the collection of execution data of their plans. To this end, GrAM uses an observation system for their behavior that includes the recording of continuous and discrete state changes caused by the robot executing its programs and by exogenous events, the control decisions made, and the beliefs and reasons for making these decisions. We call this kind of perceptual capability the collection of *experiences*. Recording experiences in this sense is the prerequisite for building grounded knowledge bases.

In GrAM, programmers can specify hierarchical hybrid automata as conceptual models of problem-solving behavior and its generation. These automata interpret the data processed by the control programs and match the data streams against the specification of the automaton. If the data stream is accepted by the automaton, then the automaton transforms it into experiences. In GrAM, RoLL (Robot Learning Language, [2]) provides mechanisms for collecting experiences which can be specified separately without requiring modifications of the control program.

Using RoLL the programmer can define a hybrid automaton that mirrors his conceptualization of a pick-and-place activity. Here the the activity consists of a goal pursuing activity (*perform*) and a failure handling state called *recover*. The *perform* state has substates namely *entity-picked-up(e1)* and *entity-put-down(e1,epos)*. The automaton is anchored into the control program by associating state jumps with labeled statements in the control programs. This way the observing automaton jumps to the successor node whenever the respective labeled statement in the program gets executed. Labeling a statement in the control program makes the respective stack frame accessible. Thus, the automaton can access and record any variable value that is visible in the respective program context. For example, the hybrid

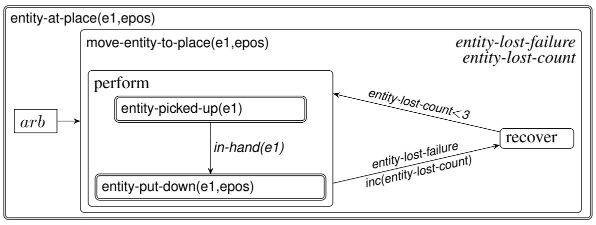

Fig. 8. Hybrid automaton for observing pick-and-place activities.

automaton can record the estimated pose of the entity to be picked up, the robot position and orientation, and the time instant upon the activation of the control routine *pick-up-entity(e1)*. During the execution of *pick-up-entity(e1)* it can record the trajectory of the arm and recognize the success of the routine.

These recorded experiences are then stored into a database that provides GrAM with the data that the knowledge chunks can be grounded in as shown in our example. GrAM provides access to the data through user-defined PROLOG predicates that we further explain in the next section.

C. Model Learning

The final part of our system is the mechanism to automatically build a classifier if a GrAM-concept is queried.

The general structure of GrAM-concepts and relations between actions, agents, objects and places needed for the learning process are stored in the GrAM ontology. Modeling this information in a general way allows to mark concepts as learnable by simply deriving them from the respective classes in the GrAM ontology.

As explained above, learning problems can be stated as intensional models which contain a set of observable and predictable properties that can be used to train a classifier to decide if an entity belongs to the modeled concept. This step is performed by the predicate *calcPlaceModel* that calculates an extensional model from an intensional one.

```
tell calcPlaceModel(IntensionalModel, ExtensionalModel) :-
       setof(O, observable(IntensionalModel, O), Observables),
       setof(P, predictable(IntensionalModel, P), Predictables),
       forPlace(IntensionalModel, Places),

       getObsValues(Observables, Places, ObservableValues),
       getPredValues(Predictables, Places, PredictableValues),

       createClassifier(Observables, ObservableValues,
                        Predictables,PredictableValues, Classifier),
       ExtensionalModel = Classifier-Observables-Predictables.
```

The first three lines determine which values are to be used for the classifier, the following two lines get their values, and the final statements train the classifier.

To get the required training data from the database, *computable* classes and properties are used. These concepts can directly be obtained from the database (*SQLComputable*) or calculated from parameters (*JythonComputable*). The system determines their values by executing the associated script or the specified SQL command. This mechanism allows for transparently retrieving external data and using it in the reasoning process.

A computable class that reads the actions for a place from the database is shown below:

```
tell action(Place, Action) :-
       db_connect(db,user,password,LinkID),
       db_query("SELECT action FROM manip_places
                 WHERE place='"&Place&"'", LinkID, Action).
```

Currently, we use decision and model trees as classifiers when learning models, but our approach is in no way limited to these algorithms. They have the advantage that the result is represented in an explicit way, so that it can easily be transformed to rules and asserted to the knowledge base. If this is not the case, however, the classifier can still be used as a "black box" that decides for each entity separately if it belongs to the represented concept or not.

IV. Related Work

A general definition of the anchoring problem, i.e. the linking between percepts and associated symbolic concepts, is elaborated by [3]. They present an approach to maintain this link, but do not go into details how it is established. [4] build a "multi-hierarchical map" combining a spatial and a semantic hierarchy based on the anchoring approach presented by Coradeschi and Saffiotti, but also avoid describing how to determine which concept a percept should be linked to.

[1] propose an approach technically similar to ours for the automatic acquisition of grounded action models. However, their work is focused on the automatic analysis of football games instead of mobile robot manipulation actions.

A system that builds grounded situation models in a block-world scenario combining continuous, stochastic and categorical information is presented by [5]. Concepts on the categorical level can be found in and created from the stochastic representation and are thus grounded in data. The setting is very simplified though, containing only few places, categories and objects.

Modayil and Kuipers present a system to automatically extract objects from an otherwise static scene by analysing differences to a laser scanner occupancy grid map [6] and to autonomously learn which actions can be used to interact with these objects [7]. Actions are learned by motor babbling, the observation of the outcome and the construction of a model. Since their robot does not have any manipulation skills, the actions are limited to pushing objects around.

V. Conclusions

We consider it very important for a robot that has to perform complex tasks in an environment together with humans to have a knowledge base that is grounded in its perception and action system.

In this paper, we present an approach for creating a practical knowledge representation for mobile robots performing manipulation actions in a household environment. We combine the Cyc upper ontology with action-related concepts that describe relations needed for mobile manipulation tasks. These concepts are described by a set of observable properties that can be used to build a model for the concept.

Our robot is equipped with methods to observe actions and their context and store the experiences in a database. These experiences can then be used to transform the general descriptions of action-related concepts into models that are grounded in actual data. This is done by training a classifier that extracts the relations between observable and predictable properties.

We implemented this system by extending PROLOG with two kinds of concepts for (a) building a model based on the intensional specification and the collected data and (b) deciding if an entity belongs to such a concept. There is no difference in the reasoning process between usual concepts and those learned from data, so they can transparently be used to define more complex concepts.

Specifying objects, places and other entities as action-related concepts, as we propose, is a very natural way of defining them, rather than describing what they look like: When the robot searches for an item, it directly knows what this item has to serve for, while it is normally not interested in its appearance.

To our knowledge, there exists no other system that is capable of learning grounded models from collected experiences and seamlessly integrates these concepts into its knowledge base. Only like that, the robot can both communicate about its actions and find concepts from its knowledge base in its sensor data.

As the next step, we plan to integrate the learned concepts into our planning system so that the robot is able to adapt its behaviour based on its experiences. Our robot is already able to transform its plans in order to optimize its actions. Learned action-related concepts would improve these transformations and allow to specificly react on objects' properties. Possible changes in the plan may be the choice of a more careful pick up strategy or a limited acceleration if a cup is filled with liquids.

VI. Acknowledgments

This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence.

References

[1] N. v. Hoyningen-Huene, B. Kirchlechner, and M. Beetz, "Gram: Reasoning with grounded action models by combining knowledge representation and data mining," in *Towards Affordance-based Robot Control*, 2007.

[2] A. Kirsch and M. Beetz, "Training on the job — collecting experience with hierarchical hybrid automata," in *Proc. KI-2007*, J. Hertzberg, M. Beetz, and R. Englert, Eds., 2007, pp. 473–476.

[3] S. Coradeschi and A. Saffiotti, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2-3, pp. 85–96, 2003.

[4] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, and J. González, "Multi-hierarchical semantic maps for mobile robotics," in *Proc. IROS 2005*, Edmonton, CA, 2005, pp. 3492–3497.

[5] N. Mavridis and D. Roy, "Grounded Situation Models for Robots: Where words and percepts meet," in *Proc. IROS 2006*, 2006, pp. 4690–4697.

[6] J. Modayil and B. Kuipers, "Autonomous Development of a Grounded Object Ontology by a Learning Robot," in *Proc. AAAI-07*, 2007, pp. 1095–1101.

[7] ——, "Where do actions come from? Autonomous robot learning of objects and actions," in *AAAI Spring Symposium*, 2007, pp. 1095–1101.