The RoboEarth Language: Representing and Exchanging Knowledge about Actions, Objects, and Environments

Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz and Michael Beetz
Department of Informatics, Technische Universität München
{tenorth,perzylo,lafrenz,beetz}@cs.tum.edu

Abstract

The community-based generation of content has been tremendously successful in the World Wide Web – people help each other by providing information that could be useful to others. We are trying to transfer this approach to robotics in order to help robots acquire the vast amounts of knowledge needed to competently perform everyday tasks. RoboEarth is intended to be a web community by robots for robots to autonomously share descriptions of tasks they have learned, object models they have created, and environments they have explored. In this paper, we report on the formal language we developed for encoding this information and present our approaches to solve the inference problems related to finding information, to determining if information is usable by a robot, and to grounding it on the robot platform.

1 Introduction

The Web 2.0 has changed the way how web content is generated. Instead of professional content providers, it is now often the users who fill web sites with text and images, forming a community of people helping each other by providing information they consider useful to others. The free encyclopedia Wikipedia grew up to millions of articles, sites like *cooking.com* or *epicurious.com* collect tens of thousands of cooking recipes, and *ehow.com* and *wikihow.com* offer instructions for all kinds of everyday tasks. "Crowdsourcing" the generation of web sites made it possible to create much more content in shorter time with shared effort.

In our research, we are trying to make use of this idea to improve the performance of our robots. On the one hand, we are working towards enabling the robots to use the large amount of information that can already be found on the Web to accomplish their tasks, for instance by translating written instructions from web pages into robot plans [Tenorth, Nyga, and Beetz, 2010]. On the other hand, we are working towards establishing a similar "World Wide Web for Robots", a web-based community in which robots can exchange knowledge *among each other*. Understanding information intended for humans is still challenging and rather costly, but once a robot has done it, it can share this newly gained information with other robots, which then do not have to go through the difficult conversion process again. We aim to speed up the time-consuming knowledge acquisition process by enabling robots to profit from tasks other robots have already learned, object models they have created, and environments they have explored.

If such information is to be autonomously generated and used by robots, that is, without human intervention, it has to be represented in a machine-understandable format. In this respect, we have much in common with the Semantic Web [Lee, Hendler, and Lassila, 2001], in which computers exchange information between each other: The meaning of the content needs to be represented explicitly, separately from platform-specific aspects, it has to be described in terms of logical axioms that a computer can understand, and these logical axioms need to be well-defined, for example in an ontology. Such an explicit representation of the semantics is important to enable a robot to *understand* the content, i.e. to set different pieces of information into relation. Only when it knows the semantics of the exchanged information, a robot can decide if an object model can be useful to perform a given task, can choose a model among different alternatives, and determine if it has all required sensors for using it.

In this paper, we describe our approach to creating a semantic representation language for the RoboEarth system. The main contributions are (1) a semantic representation language for actions, objects, and environments; (2) the infrastructure for using this representation to reason about the applicability of information in a given context and to check if all required robot capabilities are available; (3) mechanisms for creating and uploading shared knowledge. These technical contributions are validated by an experiment on a physical robot that performed a serving task based on information retrieved using the described methods. In particular, the representation language provides techniques for:

- Representation of actions and their parameters, positions of objects in the environment, and object recognition models
- Meta-information about the data to be exchanged (types, file formats, units of measure, coordinate frames)
- Requirements on components a robot needs to have in order to make use of a piece of information

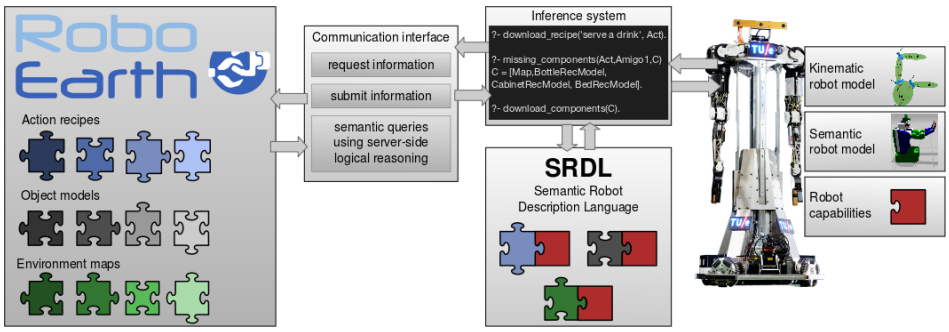

Figure 2: Overview of the RoboEarth system: A central database provides information about actions, objects, and environments. The robot can up- and download information and determine if it can use it based on a semantic model of its own capabilities.

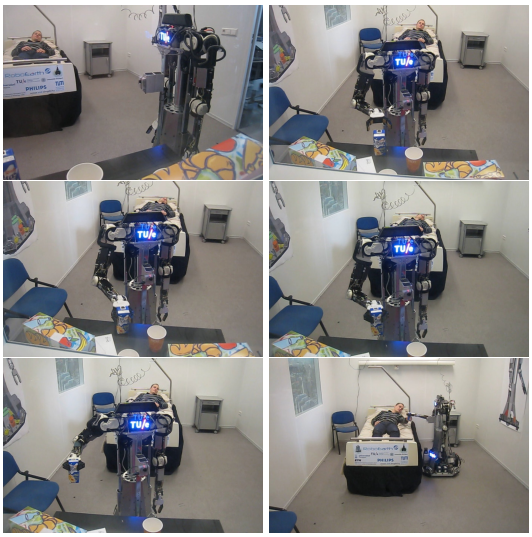

Figure 1: The Amigo robot performs a mobile manipulation task based on task descriptions, object recognition models and a semantic environment map which it has autonomously downloaded from the RoboEarth database.

- Robot self-models that describe the robot's configuration and capabilities
- Methods for matching requirement specifications to a robot's capabilities to identify missing components

RoboEarth has high demands regarding the expressiveness and the level of semantic annotations of the exchanged information: First, the platform is to be used by heterogeneous robots, i.e. no assumptions about available capabilities can be made. Second, the robots shall exchange the information autonomously without human intervention, which means that the language has to be expressive enough to provide the robot with all information needed to select information, adapt it, and reason about its applicability. These two aspects have not been tackled in prior work, which focused on the representation of actions or objects, but not on additional information that is needed for the autonomous exchange of information. Hierarchical Task Networks (HTN [Erol, Hendler, and Nau, 1994]) and related plan languages are similar to the action representation used in RoboEarth, but focus on the description of the task itself, i.e. its sub-actions, goals, and ordering constraints. XABSL [Loetzsch, Risler, and Jüngel, 2006], mainly used in the RoboCup soccer context, describes actions in terms of hierarchical finite state machines. AutomationML [Drath et al., 2008] is a standard for describing task information and spatial configurations, mainly used in industrial applications. The FIPA [O'Brien and Nicol, 1998] standard primary deals with the definition of communication standards for software agents. Object description formats like the proprietary DXF [Rudolph, Stürznickel, and Weissenberger, 1993] or the open Collada [Arnaud and Barnes, 2006] standard describe objects using meshes and textures, but without further specifying semantic properties. We are not aware of any other system that integrates task descriptions with spatial information, semantic information about object types, and meta-information about the exchanged data.

The rest of the paper is organized as follows: We start with an overview of the RoboEarth system and briefly introduce an example scenario that is used over the course of the following actions. We then describe the representations of actions, object models, and semantic environment maps, before we elaborate on the matching between action requirements and robot capabilities. Afterwards, we explain how the robot communicates with the RoboEarth database, and how it makes use of the downloaded information during task execution. We finish with a description of the experiments we performed and a discussion of the system's capabilities.

2 The RoboEarth system

This work presented in this article is part of the RoboEarth project [Waibel et al., 2011] which targets at building a "World Wide Web for Robots". Like Wikipedia, RoboEarth is to provide a platform for sharing knowledge about actions, objects, and environments between robots. The project covers different aspects like the generation and execution of task descriptions, object recognition methods, learning, the realization of the central web-based knowledge store. Parts of RoboEarth have been released as ROS packages[1]. In this paper, we focus on the methods for representing and reasoning

[1] Available at http://www.ros.org/wiki/roboearth

about the exchanged knowledge (available in the packages *re_comm* and *re_ontology*).

Figure 2 illustrates how knowledge can be exchanged via RoboEarth: On the left is the central RoboEarth knowledge base, containing descriptions of actions (called "action recipes"), objects, and environments. These pieces of information are provided by different robots with different sensing, acting and processing capabilities. Therefore, all of them have different requirements on capabilities a robot must have in order to use them, visualized by the differently shaped jigsaw pieces. The RoboEarth language thus provides methods for explicitly describing these required capabilities and for matching them against capabilities available on the robot. Each robot has a self-model consisting of a description of its *kinematic structure*, including the positions of sensors and actuators, a *semantic model* that describes the meaning of the robot's parts (e.g. that certain joints form a gripper), and a set of *software components* like object recognition systems. We use the Semantic Robot Description Language (SRDL [Kunze, Roehm, and Beetz, 2011]) to describe these components and the capabilities they provide, and to match them against the requirements specified for action recipes. Section 6 explains the process in more detail. The robot can query the RoboEarth knowledge base using interface methods that perform information encoding and communication (see Section 7).

The representation language is realized as an extension of the KNOWROB [Tenorth and Beetz, 2009] knowledge base, which we also use for grounding the downloaded descriptions on the robot (Section 8). In KNOWROB, knowledge is described in Description Logic using the Web Ontology Language (OWL). OWL distinguishes between classes, instances of these classes, and properties that can either be described for single instances or for whole classes of things. Classes are arranged in a hierarchical structure, called an ontology, allowing multiple inheritance. KNOWROB's ontology is derived from the OpenCyc ontology [Matuszek et al., 2006] and itself serves as the basis for the RoboEarth ontology. We extended the KNOWROB ontology with concepts that are especially required for the exchange of knowledge: Meta-information about the data to be exchanged like units, coordinate systems, its resolution, algorithms that were used for creating data, and requirements that are needed for interpreting it.

For the sake of clarity, we will present most of the language constructs in terms of graphical visualizations instead of source code. A more detailed description of the language capabilities can be found in a related technology report [Tenorth and Beetz, 2010], a formal specification of the language elements in the ontology at http://ias.cs.tum.edu/kb/roboearth.owl.

In the following sections, we will explain the operation on an example task on which the system's capabilities have been demonstrated: serving a drink to a patient in bed in a hospital room. The robot first needs to find a bottle in the environment, pick it up, move to the patient and hand over the bottle.

3 Actions and tasks

Action specifications, called "recipes", are described in terms of classes of actions that are arranged in a taxonomic structure. Figure 3 shows a small excerpt; in total, there are currently about 130 action classes. Users of the system can easily extend the set of actions by deriving new action classes from the existing ones. These classes form the vocabulary for describing actions, and each of them can be described by a set of properties: For instance, an action of type *Movement-TranslationEvent* can have the properties *fromLocation* and *toLocation*. Such a specification of classes by their properties is called a "class restriction" in OWL.

Figure 3: Small excerpt from the RoboEarth action ontology that describes different kinds of actions in terms of a taxonomic structure.

Actions can be arranged in a hierarchy describing the composition of complex actions from more basic ones, in addition to the generalization hierarchy in Figure 3. As an example, the action *PuttingSomethingSomewhere* for transporting an object from one position to another involves picking up an object, moving to the goal position, and putting the object down again. These sub-actions are described in the following OWL code example:

```
Class: PuttingSomethingSomewhere
  SubClassOf:
      Movement-TranslationEvent
      TransportationEvent
      subAction some PickingUpAnObject
      subAction some CarryingWhileLocomoting
      subAction some PuttingDownAnObject
      orderingConstraints value SubEventOrdering1
      orderingConstraints value SubEventOrdering2
```

The ordering of *subActions* in a task can be specified by partial ordering constraints which describe the relative pair-wise ordering between the actions.

```
Individual: SubEventOrdering1
  Types:
      PartialOrdering-Strict
  Facts:
      occursBeforeInOrdering PickingUpAnObject
      occursAfterInOrdering  CarryingWhileLocomoting

Individual: SubEventOrdering2
  Types:
      PartialOrdering-Strict
  Facts:
      occursBeforeInOrdering CarryingWhileLocomoting
      occursAfterInOrdering  PuttingDownAnObject
```

Figure 4 visualizes an action recipe for serving a drink to a patient in bed. In this picture, action classes are represented as blocks, properties of these classes are listed inside the block, and ordering constraints among the actions are shown as arrows between the blocks. There are three levels of hierarchy: The recipe for the *ServeADrink* action includes the *GraspBottle* action that, by itself, is defined by an action recipe (shown on the right side) consisting of single actions. Both recipes consist of a sequence of actions that are described as task-specific subclasses of generic actions, like *Reaching* or *Translation*, with additional parameters, like the *toLocation* or the *objectActedOn*.

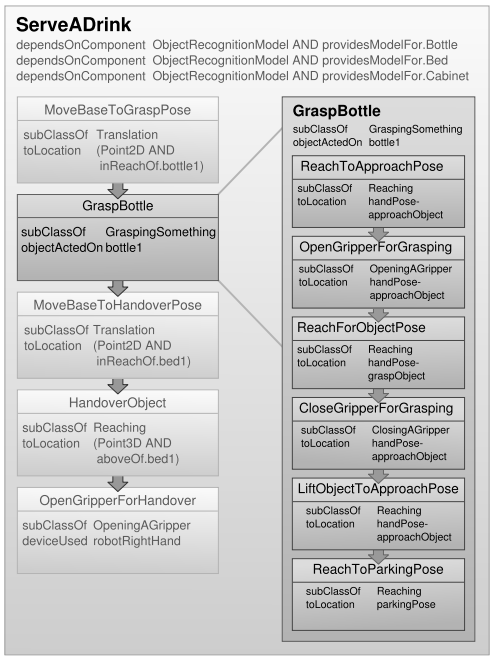

Figure 4: Representation of a "serving a drink" task, called "action recipe" in the RoboEarth terminology, which is composed of five sub-actions that themselves can be described by another action recipe.

The action recipe further lists dependencies on components that have to be available on the robot in order to successfully perform the task. In this example, there is a list of object recognition models that are necessary for the robot to recognize all objects involved in the task. There are additional dependencies inherited from higher-level action classes: *Reaching*, for example, depends on an arm motion capability, *Translation* actions require a moving base and navigation capabilities.

4 Object models

Object models in RoboEarth describe classes of objects by their semantic properties, including information on how to recognize and how to articulate them. Figure 5 exemplarily shows a model of a cabinet in a hospital room. In the upper part, there is an instance of an object recognition model that includes links to pictures and a CAD model file as well as information about the creation time and the algorithms the model can be used with. The ontology already provides class descriptions for several thousands of objects, but can easily be extended by the user – a process that can also be automated using information from the Internet [Pangercic, Haltakov, and Beetz, 2011].

This model refers to a description of the object *IkeaExpeditShelf2x2* which has articulated parts, namely doors connected to the frame via hinges. By estimating the kinematics of these doors [Sturm, Stachniss, and Burgard, 2011], the robot can determine the poses of the hinges with respect to the body of the cabinet. These poses are stored using an object-

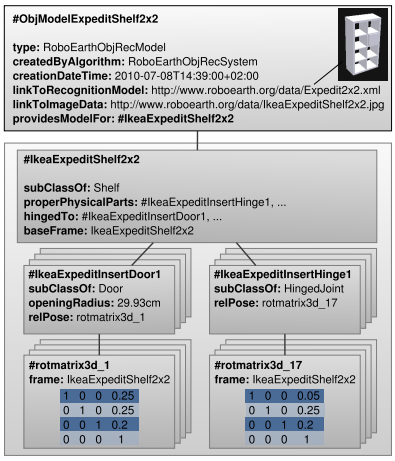

Figure 5: Object model representation. The object model instance refers to binary data for a model as well as to a detailed description of the object class to be exchanged. In this case, the model describes a cabinet composed of several articulated doors connected with hinges to the cabinet's frame.

internal coordinate system so that the information can also be applied to a different cabinet of the same type. If such a cabinet has been recognized using the model *ObjModelExpeditShelf2x2*, the relative coordinates are transformed into global map coordinates based on the pose where the cabinet body was detected. In addition to this explicit representation of coordinate frames, all numeric values can be annotated with the unit of measure provided by the extensive QUDT ontology[2]. If a user specifies the unit the results are to be returned in, values in compatible units are transparently converted, e.g. lengths from meters to feet.

The set of object models the robot can currently recognize is explicitly represented in the robot's knowledge base and can be used to decide if some kind of object can be recognized. If not, the robot can download a suitable model from RoboEarth, add its OWL description to its knowledge base, and send the recognition model (in the above example the linked CAD model file) to the perception system.

5 Environment models

There are various kinds of environment maps (topological and metric maps, two- and three-dimensional maps, maps created using different sensors like 2D laser scanners, tilting lasers or cameras, etc) that can be exchanged via RoboEarth. The language provides several classes to describe the types and properties of environment maps. Maps like occupancy grids are usually exchanged as a "black box": The robot knows which kind of map it is and which environment is described, but cannot perform reasoning about the content of the map. "Semantic maps", which consist of localized object instances (Figure 6), are a different case: They can be completely described in OWL, loaded into the knowledge base and reasoned about. The robot can for instance use such a

[2] http://qudt.org/

map to locate objects described in a recipe, or update the map with other objects it has recognized. If there are multiple maps describing the same environment, the system retrieves all of them.

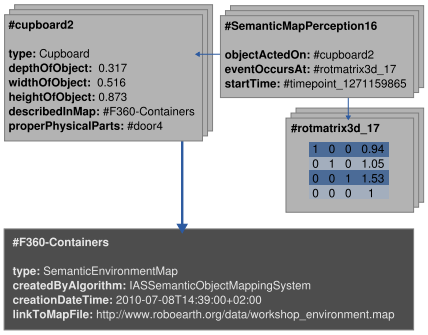

Figure 6: Environment model representation. The map *F360-Containers* links a binary file for the localization map and further contains several objects detected in the environment. Object poses are linked via a *SemanticMapPerception* instance to be able to describe poses that change over time.

6 Matching requirements to capabilities

In order to find out if the robot is able to execute a recipe and, if not, whether it can *be enabled* to do so by downloading additional information, the system matches the requirements of the action recipe to the robot's capability model. Although this procedure cannot guarantee successful task execution, the robot can determine whether something is definitely missing and if that missing part can be provided by RoboEarth.

The matching process is realized using the Semantic Robot Description Language (SRDL [Kunze, Roehm, and Beetz, 2011]) and visualized in Figure 7. The robot first queries for an action recipe and, together with the query, sends a description of its own capabilities to the inference engine, which then checks whether all requirements of the recipe are available on the robot. At first sight, the robot may find the *EnvironmentMap* to be missing, as it is neither available on the robot nor in the RoboEarth database. Knowing that both a *2DLaserScannerMap* and a *3DLaserScannerMap* are specializations of an *EnvironmentMap*, the system can infer that they can be used to fulfill the dependency. It recursively checks their dependencies and finally selects the *2DLaserScannerMap* as its dependencies are available on the robot. The matching process is continued recursively until the system finds a combination of action recipes, object- and environment models that fits the robot and does not have any unmet dependencies. This example is very simplified in that it only requires knowledge about the sub-class hierarchy, while the matching procedure also supports checking other properties of or relations between objects – everything that can be expressed in terms of OWL class restrictions.

7 Interface to the RoboEarth database

Once the missing pieces of information have been determined, they can be searched in the RoboEarth knowledge base. A communication module provides methods for up- and downloading information using HTTP requests and encapsulates the communication with the web-based RoboEarth knowledge base. The communication package further provides methods to update existing knowledge, for instance an environmental map with updated object positions or an improved action recipe. There are different possibilities to send queries to the knowledge base: If the identifier of an action recipe, object model or environment map is known, e.g. because another recipe refers to it, this item can directly be accessed. Otherwise, queries are send as a logical specification of the properties a recipe or model needs to have. For example, the robot may search for a recipe that describes a *Serving* task with a *Bottle* as *objectActedOn*, and get all recipes for specializations of such a task as result.

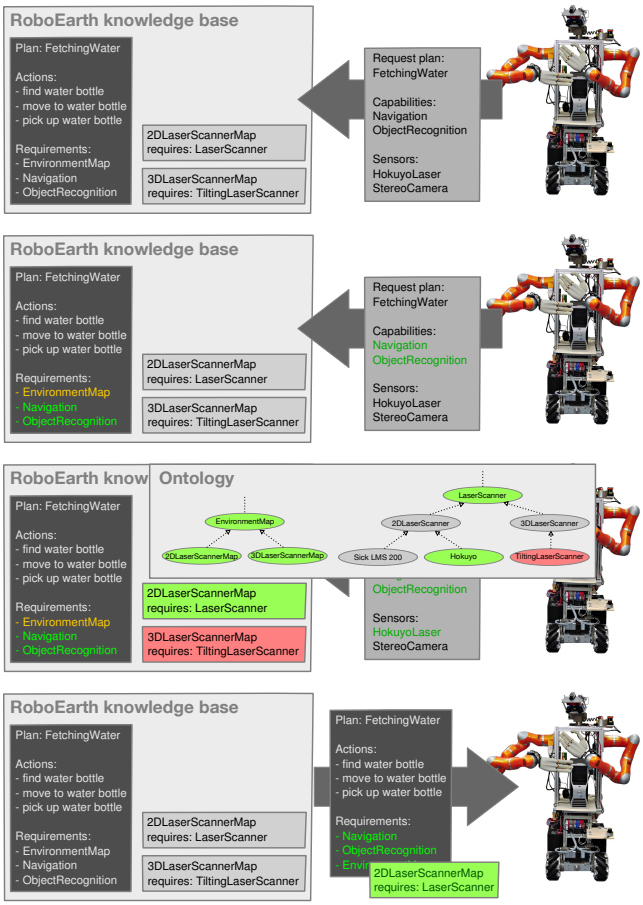

Figure 7: Matching requirements of action recipes against robot capabilities to determine which further information is still missing and has to be downloaded. The matching becomes more flexible by taking the robot's knowledge into account and selecting the *2DLaserScannerMap* to fulfill the general requirement on an *EnvironmentMap*.

8 Executing action recipes

Having downloaded information from RoboEarth, the robot has to ground the abstractly specified instructions. The action recipes need to be translated into calls to executable program code. Similar to HTN planning, which distinguishes primitive and compound tasks, we also de-compose complex tasks into more and more basic actions until we reach the level at which the actions are available as executable primitives. There is intentionally no fixed level of granularity at which this transition takes place. The system thus supports large, monolithic implementations of functionality, in which the threshold between primitives and recipes is at a rather high level, as well as setups with a large number of small components. Due to the hierarchical structure of recipes, the same high-level recipe can be executed in different setups by downloading more detailed action recipe until all actions in the recipe are available as primitives on the robot.

There are currently two options for executing action recipes: First, the rather simple RoboEarth executive component can be used, which uses mappings from action classes to action primitives that are specified using the *providedByMotionPrimitive* property. For the scenario in the experiment, only three primitives were needed: *move_gripper*, *open_gripper* and *navigate*. The second option is an exporter that creates action plans in the CRAM plan language (CPL, [Beetz, Mösenlechner, and Tenorth, 2010]) which can be executed by the CRAM executive. This allows to profit from CRAM's sophisticated techniques for failure monitoring and recovery, and especially for determining action parameters like locations where objects shall be placed. Such locations are usually described using abstract specifications like 'in reach of the patient', and need to be translated into actual metric positions. This conversion requires sophisticated reasoning methods that include geometric and kinematic information, which are provided by the CRAM system [Mösenlechner and Beetz, 2011].

9 Evaluation

This paper describes a system for representing, exchanging and reasoning about high-level task descriptions, object models, and semantic environment maps in a common semantic framework. A quantitative evaluation of such a system is hardly possible: Most task-related performance measures, like the execution time, rather describe the performance of external factors like the hardware of the executing robots than the representation language. However, the system can be evaluated on qualitative criteria: Is the representation expressive enough to encode all important kinds of information? Are all of the necessary reasoning tasks supported? Which level of autonomy can be reached?

In a recent experiment, we have demonstrated that the language and inference methods can be used to exchange information about mobile manipulation tasks. This experiment was implemented and successfully demonstrated to the public during a workshop of the RoboEarth project in Eindhoven in January 2011. Figure 1 shows the course of actions performed by the robot. A video of the experiment can be found at http://www.youtube.com/watch?v=RUJrZJyqftU.

The task the robot performed was to serve a drink to a patient in a hospital room. During the experiment, the robot downloaded the action recipe and matched it against its capabilities to determine which pieces of information are missing (as described in Section 6). It downloaded these missing components from RoboEarth, namely recognition models for all objects in the task, and a semantic environment map of the hospital room. During execution, the robot perceived the objects using the downloaded models and thereby grounded the symbols described in the recipe in object perceptions in the environment.

The experiment shows that the system is able to encode the information required for mobile pick-and-place tasks. The Amigo robot completely autonomously downloaded the action recipe, determined which information was missing, additionally downloaded these pieces of information, and thereby enabled itself to perform the task. All these reasoning tasks were performed autonomously using the methods described in this paper.

10 Discussion and conclusions

In this paper, we discussed the requirements to a formal language for representing robot knowledge with the intention of exchanging information, and presented our approach to realizing such a language. The language allows to describe actions, object recognition and articulation models, as well as semantic environment maps, and provides methods to reason about these pieces of information. Using the language, robots can autonomously decide if they lack important capabilities to perform an action, and if so, see whether they can download software that enables them to do that.

The language and the accompanying reasoning methods have successfully been used to exchange tasks, object models, and environment information on a physical mobile manipulation robot and execute the abstractly described task. This experiment showed that the presented methods enable a robot to download the information needed to perform a mobile manipulation task, including descriptions of the actions to perform, models of the objects to manipulate, and a description of the environment, from the RoboEarth database on the Internet. The current system is still limited regarding the exchange of low-level information like motion trajectories, forces or accelerations. Currently, they need to be determined by the downloading robot using motion planning components based on symbolic constraints in the recipe, which can be a limiting factor for actions that require dexterous manipulation.

Acknowledgments

This work is supported in part by the EU FP7 Project *RoboEarth* (grant number 248942).

References

[Arnaud and Barnes, 2006] Arnaud, R. and M.C. Barnes (2006). *COLLADA: sailing the gulf of 3D digital content creation*. AK Peters, Ltd.

[Beetz, Mösenlechner, and Tenorth, 2010] Beetz, Michael, Lorenz Mösenlechner, and Moritz Tenorth (2010). CRAM

– A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1012–1017, Taipei, Taiwan.

[Drath et al., 2008] Drath, R., A. Luder, J. Peschke, and L. Hundt (2008). AutomationML-the glue for seamless automation engineering. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pp. 616–623. IEEE.

[Erol, Hendler, and Nau, 1994] Erol, K., J. Hendler, and D.S. Nau (1994). Htn planning: Complexity and expressivity. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 1123–1123. John Wiley & Sons LTD.

[Kunze, Roehm, and Beetz, 2011] Kunze, Lars, Tobias Roehm, and Michael Beetz (2011). Towards semantic robot description languages. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5589–5595, Shanghai, China.

[Lee, Hendler, and Lassila, 2001] Lee, T.B., J. Hendler, and O. Lassila (2001). The Semantic Web. *Scientific American* 284(5): 34–43.

[Loetzsch, Risler, and Jüngel, 2006] Loetzsch, M., M. Risler, and M. Jüngel (2006). XABSL – A Pragmatic Approach to Behavior Engineering. In *Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pp. 5124–5129.

[Matuszek et al., 2006] Matuszek, C., J. Cabral, M. Witbrock, and J. DeOliveira (2006). An introduction to the syntax and content of Cyc. *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering* pp. 44–49.

[Mösenlechner and Beetz, 2011] Mösenlechner, Lorenz and Michael Beetz (2011). Parameterizing Actions to have the Appropriate Effects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA.

[O'Brien and Nicol, 1998] O'Brien, P.D. and R.C. Nicol (1998). FIPA – towards a standard for software agents. *BT Technology Journal* 16(3): 51–59.

[Pangercic, Haltakov, and Beetz, 2011] Pangercic, Dejan, Vladimir Haltakov, and Michael Beetz (2011). Fast and robust object detection in household environments using vocabulary trees with sift descriptors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, San Francisco, CA, USA.

[Rudolph, Stürznickel, and Weissenberger, 1993] Rudolph, D., T. Stürznickel, and L. Weissenberger (1993). *Der DXF-Standard*. Rossipaul.

[Sturm, Stachniss, and Burgard, 2011] Sturm, J., C. Stachniss, and W. Burgard (2011). A probabilistic framework for learning kinematic models of articulated objects. *Journal on Artificial Intelligence Research (JAIR)* 41: 477–626.

[Tenorth and Beetz, 2009] Tenorth, Moritz and Michael Beetz (2009). KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4261–4266.

[Tenorth and Beetz, 2010] Tenorth, Moritz and Michael Beetz (2010). Deliverable D5.2: The RoboEarth Language – Language Specification. Technical report D5.2, FP7-ICT-248942 RoboEarth.

[Tenorth, Nyga, and Beetz, 2010] Tenorth, Moritz, Daniel Nyga, and Michael Beetz (2010). Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1486–1491, Anchorage, AK, USA.

[Waibel et al., 2011] Waibel, Markus, Michael Beetz, Raffaello D'Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Oliver Zweigle, and René Molengraft (2011). RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine* 18(2): 69–82.