Representation and Exchange of Knowledge About Actions, Objects, and Environments in the RoboEarth Framework

Moritz Tenorth, *Member, IEEE*, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz, *Member, IEEE*

***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.**

***Note to Practitioners*—In this paper, we report on a formal language for knowledge representation that is used in the RoboEarth system, a web-based knowledge base intended to be like a "Wikipedia for robots." The objective is to enable robots to share information about how to perform actions, how to recognize and interact with objects, and where to find objects in an environment. The developed language allows to store such information in a format that supports logical inference, so that robots can for example autonomously decide if they have all prerequisites needed for performing a described action. In laboratory experiments, the system has been applied to the exchange of pick-and-place style activities between two mobile manipulation robots. We are currently extending the representation towards more fine-grained action specifications.**

***Index Terms*—Knowledge representation, knowledge exchange, service robotics.**

I. Introduction

THE Web 2.0 has changed the way how web content is generated. Instead of professional editors, it is now often the users who fill web sites with content, 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 employ this approach to improve the performance of our robots. On the one hand, we aim at enabling 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 [1]. On the other hand, we are working towards a similar "World-Wide Web for Robots," called RoboEarth (Fig. 1), that intends to create a web-based community in which robots can exchange knowledge *among each other*. Understanding information that was originally created for humans is still challenging and rather costly, but once one 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 hope to speed up the time-consuming knowledge acquisition process by enabling robots to profit from tasks other robots have already learned, from object models they have created, and from maps of 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, the approach has much in common with the Semantic Web [2], in which computers exchange information among each other: The meaning of content needs to be represented explicitly, be separated from platform-specific aspects, 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. An explicit representation of the semantics is important to enable robots to *understand* the content, i.e., to set single pieces of information into relation. Only if they know the semantics of the exchanged information, robots can decide if an object model will be useful to perform a given task, or determine if all required sensors are available. In particular, the representation language provides techniques for describing the following.

- Actions and their parameters, object poses in the environment, and object recognition models.
- Meta-information about the exchanged data, e.g., 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.

Manuscript received May 24, 2012; revised October 01, 2012; accepted December 26, 2012. This paper was recommended for publication by Associate Editor H. Tanner and Editor S. Sarma upon evaluation of the reviewers' comments. This work was supported in part by the EU FP7 Projects RoboEarth under Grant 248942 and *RoboHow* under Grant 288533, and the DFG excellence initiative research cluster *Cognition for Technical Systems (CoTeSys)*.

M. Tenorth and M. Beetz are with the Institute for Artificial Intelligence and TZI, Universität Bremen, Bremen 28359, Germany (e-mail: tenorth@cs.uni-bremen.de; beetz@cs.uni-bremen.de).

A. Perzylo and R. Lafrenz are with the Department of Informatics, Technische Universität München, München D-80333, Germany (e-mail: perzylo@cs.tum.edu; lafren @cs.tum.edu).

Digital Object Identifier 10.1109/TASE.2013.2244883

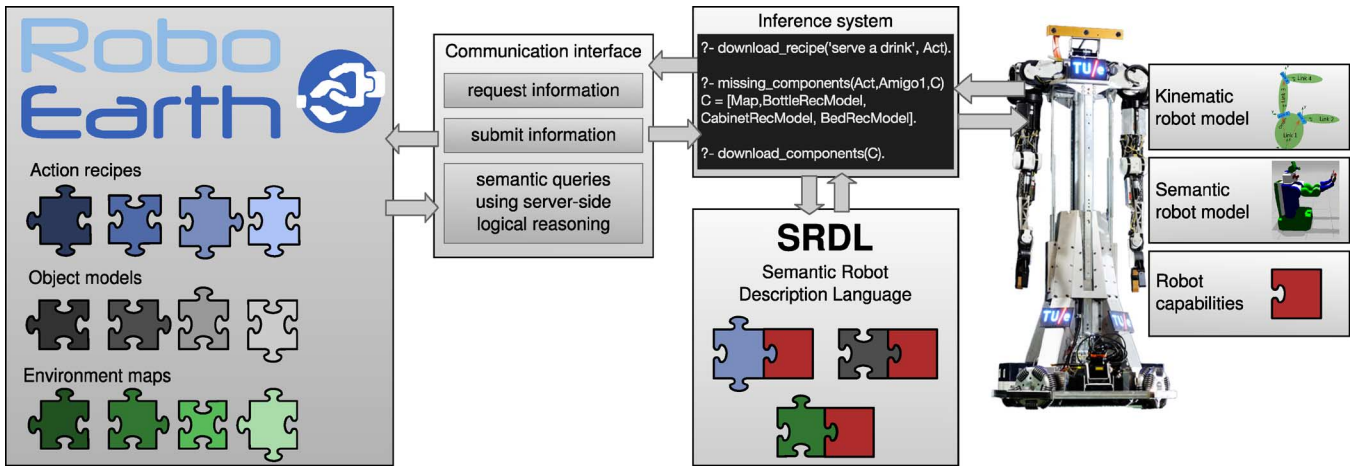

Fig. 1. Overview of the RoboEarth system: A central database provides information about actions, objects, and environments. The robot can up- and download information and determine if it can use it based on a semantic model of its own capabilities.

- Self-models of a robot's components and capability configuration.
- Methods for matching requirement specifications to a robot's capabilities to identify missing components.

In this paper, we describe our approach to creating a semantic representation language for the RoboEarth system. It is an extended and updated version of a paper presented at ICRA 2012 [3], describing more recent experiments on two heterogeneous robots in two environments as well as more detailed information about the language elements. Its main contributions are: 1) a semantic representation language for actions, objects, and environments;) 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; and 3) mechanisms for creating and uploading shared knowledge. These technical contributions are validated by an experiment including two physical robots performing a serving task in two different environments based on information retrieved using the described methods. The second robot thereby applied information the first one had shared via RoboEarth.

The rest of this paper is organized as follows. We start with an overview of related work, introduce the RoboEarth system, and describe the representations of actions, object models, and semantic environment maps. We then explain the matching between action requirements and robot capabilities, the communication with the RoboEarth database, and the application of downloaded information during task execution. We finish with a description of our experiments and a discussion of the system's capabilities.

II. Related Work

As a platform for knowledge exchange between heterogeneous robots, RoboEarth requires very expressive and highly semantic representations that provide a robot with all information it needs to select information from the knowledge base, adapt it, and reason about its applicability. Earlier research on knowledge representation for actions or objects usually did not deal with this kind of meta-information needed for autonomously exchanging knowledge. Hierarchical Task Networks (HTN [4]) and related languages for plan representation [19] or workflow specification [18] are similar to the action representation used in RoboEarth, but focus on the description of the task itself, i.e., its subactions, goals, and ordering constraints. The Planning Domain Definition Language (PDDL) [20] follows a different approach describing actions as first principles from which plans are constructed during run-time using AI planning techniques. XABSL [5], mainly used in the RoboCup soccer context, describes actions in terms of hierarchical finite state machines. AutomationML [6] is a standard for describing task information and spatial configurations, mainly used in industrial applications. The FIPA [7] standard primarily deals with the definition of communication standards for software agents. Object description formats like the proprietary DXF [8] or the open Collada [9] standard describe objects by their meshes and textures, but without further specifying semantic properties. The Knowledge Interchange Format (KIF) [21] is a very expressive generic exchange language that aims at a self-contained representation. The high expressiveness however comes at the cost of limited reasoning support. For RoboEarth, we chose a shared ontology as pragmatic solution. We are not aware of any other system that integrates task descriptions, spatial information, semantic information about object types and meta-information about the exchanged data in a common language supporting abstract reasoning.

Related work on sharing knowledge among robots focused either on sharing a common belief state in multirobot systems [10], or on fundamental aspects like how heterogeneous robots can autonomously acquire and share symbols created from perceptual cues [11]. Our interest is rather on creating a system for exchanging complex manipulation task-related information, so we simplify some of these aspects by assuming that a common base ontology is shared by all parties and that perception is done using the provided object models which are linked to the classes in the ontology.

The RoboEarth system as a whole is complemented by other efforts to create cloud-based robotic applications. Other systems investigate aspects like remote sensor data processing [12], teleoperation via the Internet [13], service-oriented interfaces for robot components [14], as well as distributed architectures for task execution and coordination [15]. Our

approach to capability representation and matching is inspired by work on web services in the semantic web context [16], [17] that face similar problems in web service discovery and composition, though precondition checks are usually rather shallow in these systems and rely on service descriptions specified by a programmer.

III. The RoboEarth System

The work presented in this paper is part of the RoboEarth project [22] which targets at building a Wikipedia-like 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, and the realization of the central web-based knowledge store. Parts of RoboEarth have been released as open-source ROS packages.[1] In this paper, we focus on the methods for representing and reasoning about the exchanged knowledge. The source code is available in the ROS packages *re_comm* and *re_ontology* and the *knowrob* stack. For technical details regarding the implementation of the presented techniques, we refer to the documentation of these software packages.

Fig. 1 illustrates how knowledge can be exchanged via the RoboEarth platform: On the left is the central RoboEarth knowledge base, containing descriptions of actions (called "action recipes"), objects, and environments. These pieces of information have been provided by different robots with different sensing, acting and processing capabilities, and therefore have different requirements on capabilities a robot must have in order to use them. The RoboEarth language provides methods for explicitly describing these required capabilities and for matching them against capabilities available on the robot, visualized by the different shapes of puzzle pieces. Each robot has a self-model consisting of a description of its *kinematic structure*, including the positions of sensors and actuators, a *semantic model* of its parts (describing, e.g., that a group of parts forms a gripper), and a set of *software components* like object recognition systems. We apply the Semantic Robot Description Language (SRDL) [23] to describe these components and the capabilities they provide, and to match them against the requirements specified for action recipes. Section VII explains the process in more detail. The robot can connect to the RoboEarth knowledge base using interface methods that perform information encoding and communication (see Section VIII).

The representation language is realized as an extension of the KnowRob [24] ontology, which is also used for grounding the downloaded descriptions on the robot (Section IX). 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 [25] 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 technical report [26]. A formal specification of the language elements is provided by the ontology[2] and the OWL files imported by *roboearth.owl*.

IV. Actions and Tasks

Actions are specified by deriving a subclass from one of the action classes in the KnowRob ontology, which currently contains about 130 action classes (Fig. 2), and extending the description of this subclass with task-specific properties. For instance, an action of type *TransportationEvent* should have the properties *fromLocation* and *toLocation* as well as an *objectActedOn*. The specification of properties a class needs to have is called a "class restriction" in OWL.

Actions can be arranged in a temporal hierarchy describing the composition of complex actions from more basic ones, in addition to the generalization hierarchy in Fig. 2. 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 subactions 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 pairwise ordering between the actions.

```
Individual: SubEventOrdering1
   Types:
      PartialOrdering-Strict
   Facts:
      occursBeforeInOrdering PickingUpAnObject
      occursAfterInOrdering CarryingWhileLocomoting
Individual: SubEventOrdering2
   Types:
      PartialOrdering-Strict
   Facts:
      occursBeforeInOrdering CarryingWhileLocomoting
      occursAfterInOrdering PuttingDownAnObject
```

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

[2] http://ias.cs.tum.edu/kb/roboearth.owl.

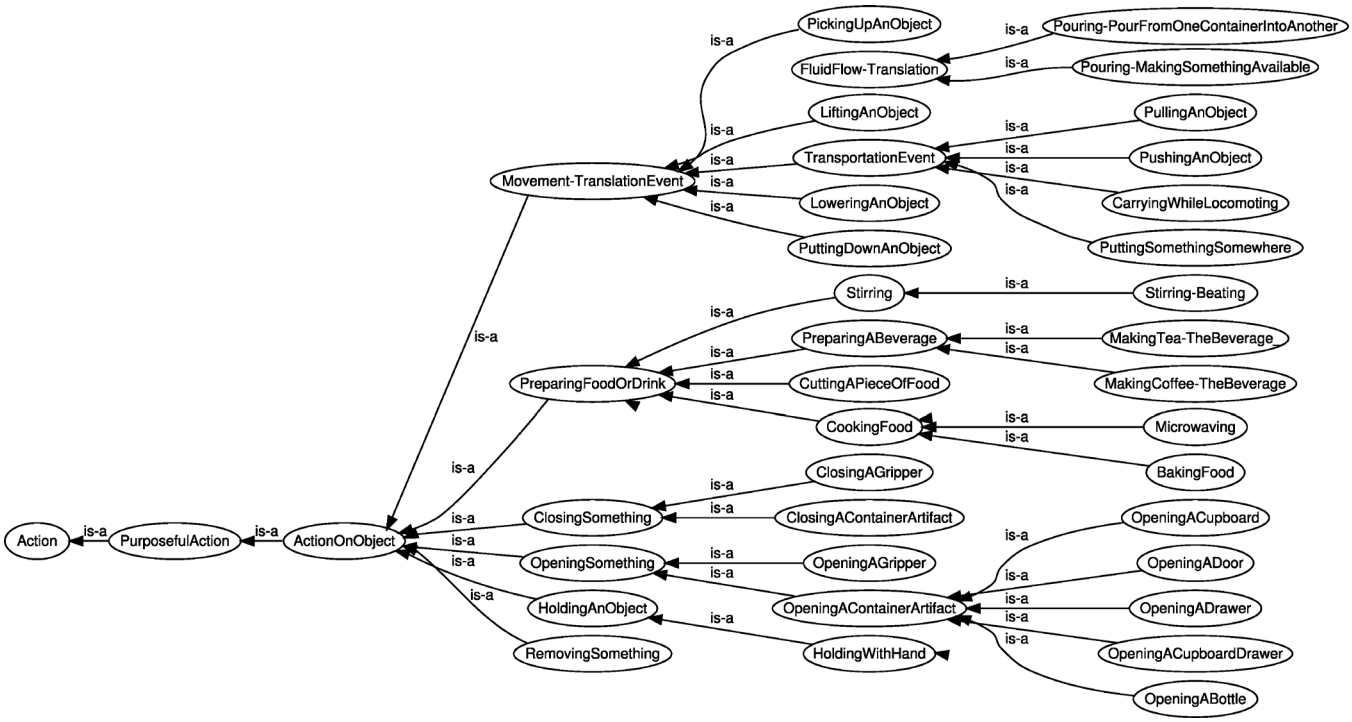

Fig. 2. Excerpt from the ROBOEARTH action ontology that describes different kinds of actions in terms of a taxonomic structure. Each of these classes is further annotated with its semantic properties. Similar ontologies exist for classes of objects and environment models.

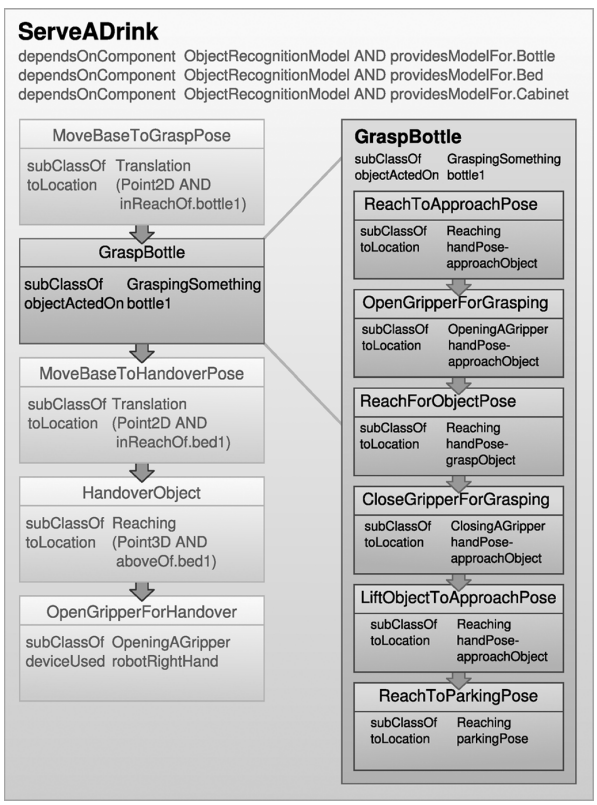

Fig. 3. Representation of a "serving a drink" task, called "action recipe" in the ROBOEARTH terminology, which is composed of five subactions that themselves can be described by another action recipe.

Fig. 3 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*. This specification can be transformed into appropriate calls to executable components for executing the task. Before execution, the abstract descriptions of objects and locations need to be grounded in concrete locations using the perception methods and the environment model (Section IX). The action recipe lists dependencies on components that have to be available on the robot in order to successfully perform the task, in this example some object recognition models that are necessary to recognize all objects in the task. Additional dependencies are inherited from higher level action classes, exploiting the hierarchical structure of the action ontology. The dependency on an arm motion capability, for example, is specified for all subclasses of *Reaching* at once and therefore does not have to be specified in each action recipe.

V. 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. Fig. 4 exemplarily shows a model of a cabinet in a mock-up hospital room. The upper part describes an instance of an object recognition model, including links to pictures and a CAD model as well as information about the creation time and the algorithm that can use the model. The recognition model instance refers to a description of the object

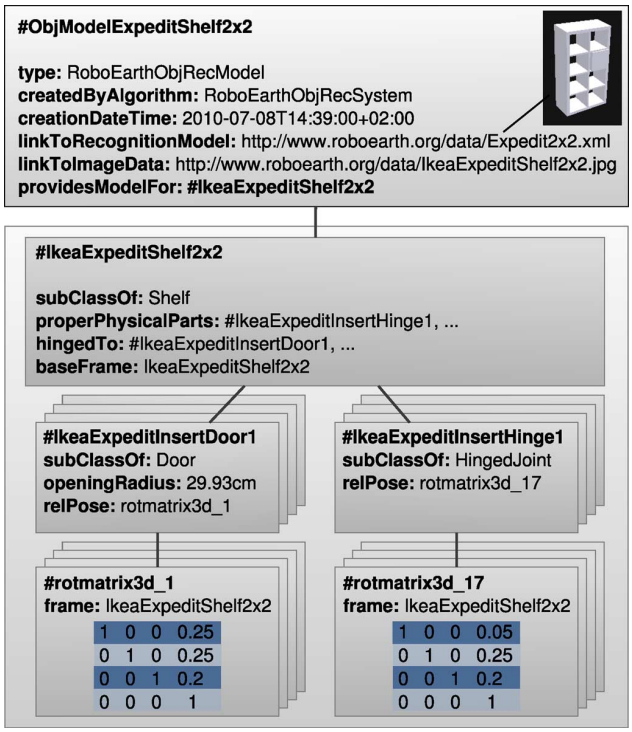

Fig. 4. Object model representation. The object model instance refers to binary data for a model as well as to a detailed description of the object class to be exchanged. In this case, the model describes a cabinet composed of several articulated doors connected with hinges to the cabinet's frame.

class *IkeaExpeditShelf2x2* that consists of articulated parts, namely, doors connected to its frame via hinges. The relative poses of the hinges with respect to the body of the cabinet can be estimated [27] and stored in the class description. This way, the information about their locations and properties can be applied to other instances of the same type of cabinet. Once the cabinet has been recognized using the model *ObjModelExpeditShelf2x2*, the relative coordinates are transformed into global coordinates based on the estimated pose of the cabinet. All coordinate frames are explicitly described, and all numeric values can be annotated with their unit of measure from the QUDT ontology[3], allowing transparent conversion of, e.g., lengths from meters to feet.

After downloading an object model, the robot sends the linked recognition model file to its perception system and loads its OWL description into its local knowledge base. Based on the set of available object models and their links to object classes, the robot can determine whether it can recognize a certain kind of object or if a model needs to be downloaded.

VI. Environment Models

RoboEarth supports various kinds of environment maps (topological and metric maps, two- and three-dimensional, created using different sensors like 2D laser scanners, tilting lasers or cameras, etc.). Fig. 5 describes their representation in the RoboEarth language: Each map is annotated with an OWL description specifying its type and some basic properties. The map content can either be described in the same OWL file, which enables the system to reason about it, or be linked

[3]http://qudt.org/.

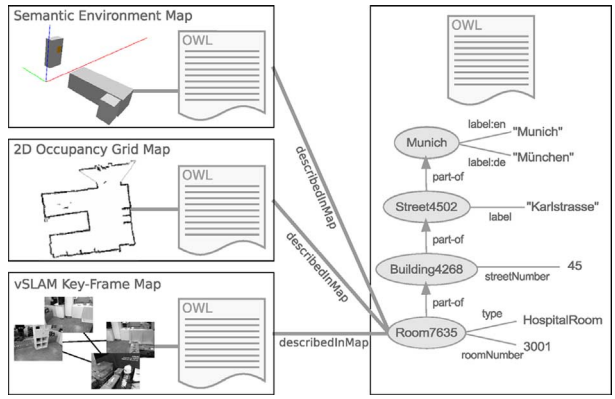

Fig. 5. Environment model representation. Different types of maps are supported and either described completely in the RoboEarth language, or in a linked binary file. A spatial hierarchy of room, building, street, city, etc. describes which environment the map belongs to and allows to search for suitable maps in RoboEarth.

as external (binary) files. The latter is often used for maps for which established file formats exist or where logical inference would not make sense (e.g., occupancy grid maps).

In order to exchange these maps, they need to be annotated with information about which environment they describe so that a robot searching for information can find them in the database. The approach chosen for the RoboEarth language is similar to the human way of describing an address: Maps are annotated with the city, street, building, floor, and the number or type of room they describe, as shown in the right side of Fig. 5. These elements are linked by a transitive *part-of* relation. This allows to query for combinations of these levels, e.g., to search for all maps of a kitchen in Karlstrasse, Munich, or for all rooms on the third floor of Karlstrasse 45. It further allows to combine labels, such as room numbers, with types of rooms (since private homes usually do not have room numbers) and to attach multiple labels to the same physical entity (Karlsplatz and Stachus are two names for the same square in Munich). This flexible representation supports both the spatial hierarchy (city—street—building), the semantic/generalization hierarchy (room—kitchen), and different labels for the same room.

VII. Matching Requirements to Capabilities

In order to find out if the robot has all prerequisites for executing a recipe or, if not, whether missing components can be downloaded from RoboEarth, the system matches the requirements of the action recipe to the robot's capability model. While this procedure cannot guarantee successful task execution, it can at least determine if components are missing and need to be retrieved. The matching process is realized using the Semantic Robot Description Language (SRDL) [23] and visualized in Fig. 6. 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. 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. Using knowledge that both a *2DLaserScannerMap* and a *3DLaserScannerMap* are specializations of an *EnvironmentMap*, it can infer that both can be used to fulfill the dependency. It recursively checks

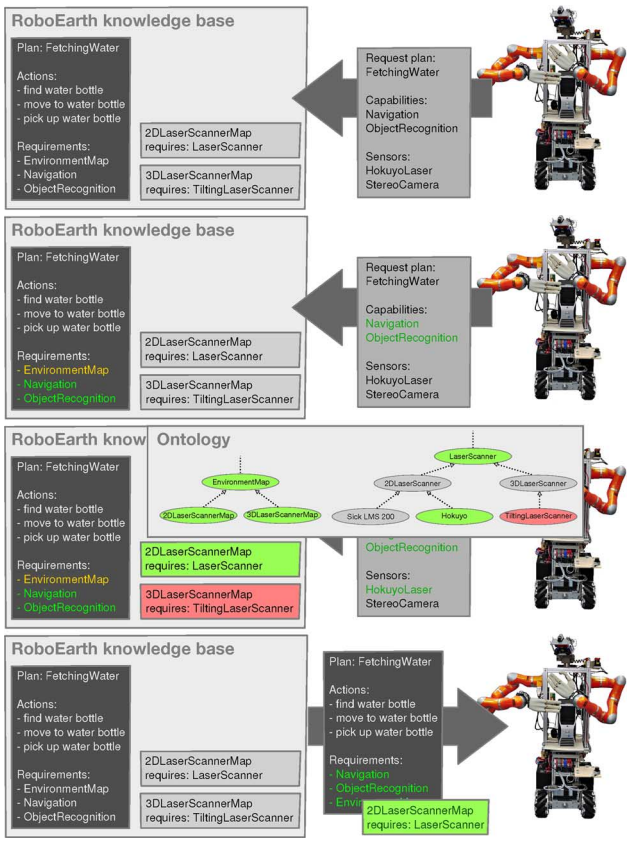

Fig. 6. Matching requirements of action recipes against robot capabilities to determine which further information is still missing and has to be downloaded. The matching becomes more flexible by taking the robot's knowledge into account and selecting the *2DLaserScannerMap* to fulfill the general requirement on an *EnvironmentMap*.

their dependencies and finally selects the *2DLaserScannerMap* whose dependencies are available on the robot. The matching process is continued until the system finds a combination of action recipes, object- and environment models that does not have any unmet dependencies, or until all alternatives are exhausted and no solution could be found.

This matching procedure is realized by a set of Prolog rules which read all available components by searching along the robot's kinematic structure (*comp_available(Robot, CompT)*), and all capabilities which are either asserted to be available on a class of robots or a robot instance, or inferred to be available because they solely depend on available components or capabilities. The following code examples have been slightly simplified, the full source code is contained in the *mod_srdl* package.[4]

```
cap_available(Cap, Robot) :-
   owl_has(Robot, hasCapability, SubCap),
   owl_subclass_of(SubCap, Cap).
cap_available(Cap, Robot) :-
   rdfs_individual_of(Robot, RobotT),
   class_properties(RobotT, hasCapability, SubCap),
   owl_subclass_of(SubCap, Cap).
cap_available(Cap, Robot) :-
   forall(class_properties(Cap, dependsOnComp,
   CompT),
      comp_available(Robot, CompT)),
   forall(class_properties(Cap, dependsOnCap,
   SubCap),
         cap_available(SubCap, Robot)).
```

The *required_comp* predicate collects the set of components that the action itself, any of its subactions, or any required components or capabilities depend on (analogous for capabilities):

```
required_comp(Act, Comp) :-
      plan_subevents_recursive(Act, SubAct),
      class_properties(SubAct, dependsOnComp, Comp).
required_comp(Act, Comp) :-
      required_cap(Act, Cap),
      class_properties(Cap, dependsOnComp, Comp).
```

Based on these rules, missing components and capabilities can be defined as "required and not available," and the feasibility of an action as "does not depend on any missing capabilities or components":

```
missing_comp(Act, Robot, Comp) :-
   required_comp(Act, Comp),
   not(comp_available(Robot, Comp)).
action_feasible_on_robot(Act, Robot) :-
   not(missing_cap(Act, Robot, _)),
   not(missing_comp(Act, Robot, _)).
```

These rules appear very simple because they make use of the specialization hierarchy (inherited dependencies) and composition hierarchy (dependencies of subactions) in the action representation as well as transitivity of the *sub_component* predicate operating on the robot's kinematic structure. While the above matching example only requires information about the asserted subclass hierarchy, the actual matching procedure supports dependency specifications that make full use of the expressiveness of OWL class restrictions which are evaluated by the *owl_individual_of* and *owl_subclass_of* predicates.

VIII. Communication With the RoboEarth Knowledge Base

Once the missing pieces of information have been determined, the robot searches for them 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 for updating existing knowledge, for instance an environmental map with updated object positions or an improved action recipe. There are different possibilities for sending 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 sent as a logical specification of the properties a recipe or model needs to have. For example,

[4]http://ros.org/wiki/mod_srdl

the robot may search for a recipe that describes a *Serving* task with a *Bottle* as *objectActedOn*, and, as a result, get all recipes describing specializations of such a task.

IX. Execution of Action Recipes

Having downloaded abstractly specified instructions from RoboEarth, the robot has to ground them in calls to executable program code. Complex tasks are decomposed into more and more basic actions until executable primitives for the actions are available. There is intentionally no predefined level of granularity at which this transition takes place to let the system support large, monolithic implementations of functionality as well as setups with a large number of small components. The same high-level recipe can be executed in different setups by downloading more fine-grained action recipes until executable primitives are available for each action.

The execution of action recipes can be realized using different techniques; system integrators can create their own execution engine interpreting the task descriptions in an action recipe. We created a reference implementation of an execution engine [28] based on the Cognitive Robot Abstract Machine framework (CRAM) [29]. In this implementation, action recipes are translated into robot plans described in the CRAM Plan Language (CPL). Compared to the OWL-based language for action recipes that is optimized for reasoning and for integrating information sources, CPL specializes on the execution of plans. The CRAM system offers sophisticated techniques for failure monitoring and recovery and for choosing suitable action parameters. In action recipes, parameters like the locations where objects are to be placed are described using abstract specifications like "in reach of the patient." CRAM provides methods for generating metric positions that comply with the abstract specification [30].

X. 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. The times needed for download and inference depend on the size and complexity of the task description and the robot model. In our experiments with the PR2 (whose model describes 158 components) and the drink-serving task, finding the recipe in the database including capability matching took about 1.41 s, its download another 1–2 s. The system can further 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 achieved? We thus investigated how the RoboEarth language can enable robots to perform tasks in a previously unknown environment. The experiment included two heterogeneous robot platforms, a PR2 and an Amigo robot, operating at two different locations that were previously unknown to them. Using information from RoboEarth, they were supposed to serve a drink to a patient in bed, which involved navigating to a cabinet, opening it, taking the drink out of it, and handing it to the patient.

Although both environments had a different spatial layout, they shared common pieces of furniture, for instance the same type of cabinet. This allowed sharing object-related information across environments: When the first robot performed the task, it did not know about the articulation properties of the cabinet door. It thus estimated the type and pose of the joint, attached this information to the object model and uploaded it to the RoboEarth knowledge base. While an object instance is environment-dependent, the model is self-contained and can therefore be applied to objects in different scenes, which allowed the second robot to open the door. Without this model, it would not have been capable of doing that since it did not have a compliant arm. The upper part of Fig. 7 shows the environment maps that were downloaded from RoboEarth. The following query was used to download the map information using the SeRQL [31] query interface[5] offered by the RoboEarth knowledge base:

```
select source from context source {R}
      kr:describedInMap {S};
      kr:roomNumber {N}
      where N like "3001"
      using namespace
      kr = ⟨http://ias.cs.tum.edu/kb/knowrob.owl#⟩;
```

Based on this map, the robots (Fig. 7 bottom) could navigate to the appropriate positions and locate the objects required for the task. The action recipe to be used was selected using the following query:

```
select source from context source {A}
   rdfs:label {"serve a drink"^^xsd:string}
   using namespace
   rdfs = ⟨http://www.w3.org/2000/01/rdf-schema#⟩
```

The action recipe was then matched against the robots' capabilities with the result that all required capabilities were available, but some recognition models for some of the objects mentioned in the task were missing (namely, the bottle and the bed), which had to be downloaded, including the CAD models shown in Fig. 7. The following CPL plan was then generated from the action recipe:

```
(def-top-level-plan serve-a-drink ()
(with-designators (
   (bottle1 (object '((name bottle1)
      (type drinking-bottle))))
   (bed1 (object '((name bed1)
      (type bed_piece-of-furniture))))
   (hand-pose-handover1 (location '((on, bed1))))
   (robot-pose-handover1 (location '((to reach)
      (side :right)
      (loc, hand-pose-handover1))))
```

[5]http://api.roboearth.org

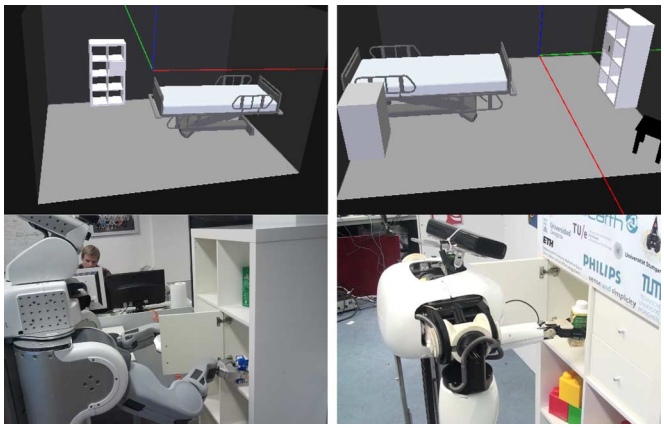

Fig. 7. Top: Semantic environment maps of the two hospital rooms, downloaded from RoboEarth based on the address and room number. Bottom: PR2 and Amigo robots opening the cabinet and picking up the drink to be served.

```
    (arms-at101 (action '((type trajectory)
       (pose, hand-pose-handover1)
       (side :right))))
    (unhand-action102 (action '((type open-gripper)
       (side :right))))
   )
    (achieve '(object-in-hand, bottle1 :right))
    (at-location (robot-pose-handover1))
    (achieve '(arms-at, arms-at101))
(achieve '(arms-at, unhand-action102))))
```

XI. Discussion

Since the first version of this paper, the described methods have been applied to other tasks and domains, for example to enable robots to interact with customers in a convenience store [32]. The implemented recommendation tasks did not involve mobile object manipulation, but other challenges like interaction with customers and question answering based on the robot's knowledge about objects and on the semantic environment map. RoboEarth has been used for exchanging the required task-, object-, and environment knowledge. Especially the behavior definition in a recipe (instead of compiled program code) and the modularity of spatial, semantic and action-related knowledge was very important in this scenario to facilitate the modification of robot behavior and deployment on several robots.

While we have so far focused on the technical realization of the representation language and reasoning modules described in this paper, there are several open research issues related to the RoboEarth vision of a World-Wide Web for robots. For example, the current language for action recipes is limited to symbolic action descriptions. Describing lower level motions, accelerations and forces would however allow the exchange of novel actions that are not available as executable modules on the target robot. One option that we are currently exploring is to include symbolic motion constraints that can be interpreted by a motion controller [33].

Once RoboEarth grows to many users, scalability will also become an issue—not only regarding the infrastructure, but also for managing the quality of the stored information: Which measures are needed to keep a large crowdsourced database structured? Which kinds of information do users actually want to share? Is the language expressive enough for all of them? Can the quality and safety of the information in the database for example be ensured by a rating system based on robot experiences? Are human moderators needed? How does a robot rank and select information if there are multiple alternatives, e.g., hundreds of models for different kinds of cups and bottles?

These research questions are not yet addressed by the current version of the system, though solutions to at least many of them will be needed for a usable and scalable crowdsourced robot knowledge base. The relevance of the individual aspects will also become clearer once first experiences in larger settings and with more robots have been collected. We hope that techniques that proved successful for human crowdsourcing can be applied in a modified form, also exploiting opportunities particular to the robot setting. For example, ratings of downloaded information can be automatically given, and robots can upload a detailed record of their experiences with executing a task.

Another interesting research opportunity that will arise once the RoboEarth system includes a substantial amount of data is learning on the database. Massive amounts of data about environments, objects and action log data could enable robots to learn typical object locations, success models of plans given the context, common execution failures, timing information, or promising plans for a given robot platform.

XII. Conclusion

In this paper, we discussed the requirements of a formal language for representing robot knowledge with the intention of exchanging it, 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 any capabilities that are needed to perform an action, and if so, see whether they can download software to acquire them. RoboEarth thereby acts as a complement, not a substitute of existing control structures: If applicable information can be found, it will help a robot with its tasks—if not, its queries will fail and it will be in the same situation as without RoboEarth.

The language and the accompanying reasoning methods have successfully been used to exchange tasks, object models, and environment maps among physical mobile manipulation robots and to execute the abstractly described task. The experiments showed that the presented methods enable robots 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.

References

[1] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz, "Web-enabled robots—Robots that use the web as an information resource," *Robot. Autom. Mag.*, vol. 18, no. 2, pp. 58–68, 2011.

[2] T. Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific Amer.*, vol. 284, no. 5, pp. 34–43, 2001.

[3] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, May 14–18, 2012, Best Cognitive Robotics Paper Award.

[4] K. Erol, J. Hendler, and D. Nau, "HTN planning: Complexity and expressivity," in *Proc. Nat. Conf. Artif. Intell.*, 1994, pp. 1123–1123.

[5] M. Loetzsch, M. Risler, and M. Jüngel, "XABSL—A pragmatic approach to behavior engineering," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2006, pp. 5124–5129.

[6] R. Drath, A. Luder, J. Peschke, and L. Hundt, "Automation ML—The glue for seamless automation engineering," in *Proc. IEEE Int. Conf. Emerging Technol. Factory Autom. (ETFA)*, 2008, pp. 616–623, IEEE.

[7] P. O'Brien and R. Nicol, "FIPA—Towards a standard for software agents," *BT Technology J.*, vol. 16, no. 3, pp. 51–59, 1998.

[8] D. Rudolph, T. Stürznickel, and L. Weissenberger*, Der DXF-Standard*. Munich, Germany: Rossipaul, 1993.

[9] R. Arnaud and M. Barnes*, COLLADA: Sailing the Gulf of 3D Digital Content Creation*. Boca Raton, FL, USA: AK Peters, Ltd., 2006.

[10] A. Khoo and I. Horswill, "Grounding inference in distributed multi-robot environments," *Robot. Autonomous Syst.*, vol. 43, no. 2, pp. 121–132, 2003.

[11] Z. Kira, "Communication and alignment of grounded symbolic knowledge among heterogeneous robots," Ph.D. dissertation, Georgia Tech, Atlanta, GA, USA, 2010.

[12] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. Kong, A. Kumar, K. Meng, and G. Kit, "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 3084–3089.

[13] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O. Jenkins, "OR2 remote lab: An environment for remote development and experimentation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 3200–3205.

[14] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in *5th IEEE Int. Symp. Service Oriented Syst. Eng. (SOSE)*, 2010, pp. 151–158.

[15] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *IEEE Network Mag.*, vol. 26, no. 3, pp. 28–34, May–Jun. 2012.

[16] W3C, W*eb Services Description Language (WSDL) Version 2.0* World Wide Web Consortium, 2007. [Online]. Available: http://www.w3.org/TR/wsdl20/

[17] H. Luokai, Y. Shi, Z. Kai, and C. Rui, "A semantic web service description language," in *Int. Conf. Inform. Eng. (ICIE)*, July 2009, vol. 2, pp. 449–452.

[18] K. L. Myers and P. M. Berry, "Workflow management systems: An AI perspective," Artificial Intelligence Center, SRI International, Menlo Park, CA, USA, Tech. Rep., 1998.

[19] K. L. Myers and D. E. Wilkins, "The ACT formalism, Version 2.2," Artificial Intelligence Center, SRI International, Menlo Park, CA, USA, Tech. Rep., 1997.

[20] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL—The planning domain definition language," *AIPS-98 Planning Committee*, 1998.

[21] M. Genesereth *et al.*, "Knowledge interchange format—Version 3.0: Reference Manual," Comput. Sci. Dept., Stanford Univ., Palo Alto, CA, USA, Tech. Rep., 1992.

[22] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Häussermann, J. Montiel, A. Perzylo, B. Schießle, O. Zweigle, and R. van de Molengraft, "RoboEarth—A world wide web for robots," *Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, 2011.

[23] L. Kunze, T. Roehm, and M. Beetz, "Towards semantic robot description languages," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 9–13, 2011, pp. 5589–5595.

[24] M. Tenorth and M. Beetz, "KnowRob—Knowledge processing for autonomous personal robots," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2009, pp. 4261–4266.

[25] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, "An introduction to the syntax and content of Cyc," in *Proc. AAAI Spring Symp. Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 2006, pp. 44–49.

[26] M. Tenorth, A. Perzylo, R. Lafrenz, M. Beetz, L. Kunze, and T. Roehm, "Deliverable D5.2-2: The RoboEarth language—Language specification," Tech. Rep. D5.2-2, 2011, FP7-ICT-248942 RoboEarth.

[27] J. Sturm, C. Stachniss, and W. Burgard, "Learning kinematic models for articulated objects," *J. Artif. Intell. Res. (JAIR)*, vol. 41, pp. 477–626, 2011.

[28] D. di Marco, M. Tenorth, K. Häussermann, O. Zweigle, and P. Levi, "Roboearth action recipe execution," in *12th Int. Conf. Intell. Autonomous Syst.*, 2012.

[29] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM—A cognitive robot abstract machine for everyday manipulation in human environments," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Taipei, Taiwan, Oct. 18–22, 2010, pp. 1012–1017.

[30] L. Mösenlechner and M. Beetz, "Parameterizing actions to have the appropriate effects," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, San Francisco, CA, USA, Sep. 25–30, 2011.

[31] J. Broekstra and A. Kampman, "SERQL: A second generation RDF query language," in *Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, 2003, pp. 13–14.

[32] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, and N. Hagita, "Towards a networked robot architecture for distributed task execution and knowledge exchange," in *3rd Int. Workshop on Standards and Common Platforms for Robot. (SCPR 2012)*, 2012.

[33] I. Kresse and M. Beetz, "Movement-aware action control—Integrating symbolic and control-theoretic action execution," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, May 14–18, 2012.

Moritz Tenorth (M'11) received the Doctoral degree in computer science from TU München, München, Germany, studied electrical engineering at RWTH Aachen and ENSTA-ParisTech and received the Diplom degree (equiv. M.Eng.) from RWTH Aachen, Germany, in 2007.

He is a Postdoctoral Researcher at the Institute for Artificial Intelligence, University of Bremen, Bremen, Germany. His research interests include grounded knowledge representations which integrate information from web sources, observed sensor data and data mining techniques, and their applications to knowledge-based action interpretation and robot control.

Alexander Perzylo received the B.Sc. and M.Sc. degrees in computer science from Technische Universität München, München, Germany.

He is a Research Assistant and the Chair of Robotics and Embedded Systems at the Technische Universität München. His research interests include knowledge representation and the generation and understanding of natural language in the context of human-robot interaction. He has a general interest in the field of service robotics.

Reinhard Lafrenz received the Diploma degree in informatics from the University of Kaiserslautern, Kaiserslautern, Germany, and the Ph.D. degree from University of Stuttgart, Stuttgart, Germany, in 2007.

He is a Senior Lecturer at Technische Universität München, München, Germany, since 2009. His research interests include perception, knowledge representation and world modeling for robots, human-robot interaction, and cognitive skills in industrial applications.

Michael Beetz (M'08) received the Diploma degree in informatics (with distinction) from the University of Kaiserslautern, Kaiserslautern, Germany, and the M.Sc., M.Phil., and Ph.D. degrees from Yale University, New Haven, CT, USA, in 1993, 1994, and 1996, and the Venia Legendi from the University of Bonn, Bonn. Germany, in 2000.

His research interests include integrated cognition-enabled robotic systems, plan-based control of autonomous robots, knowledge representation and processing for robots, integrated robot learning, and cognitive perception.