2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Towards Semantic Robot Description Languages

Lars Kunze, Tobias Roehm, and Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{kunzel, roehm, beetz}@cs.tum.edu

***Abstract*— There is a semantic gap between simple but high-level action instructions like "Pick up the cup with the right hand" and low-level robot descriptions that model, for example, the structure and kinematics of a robot's manipulator. Currently, programmers bridge this gap by mapping abstract instructions to parametrized algorithms and rigid body parts of a robot within their control programs. By linking descriptions of robot components, i.e. sensors, actuators and control programs, via capabilities to actions in an ontology we equip robots with knowledge about themselves that allows them to infer the required components for performing a given action. Thereby a robot that is instructed by an end-user, a programmer, or even another robot to perform a certain action, can assess itself whether it is able and how to perform the requested action. This self-knowledge for robots could considerably change the way of robot control, robot interaction, robot programming, and multi-robot communication.**

I. INTRODUCTION

Increasing the re-usability of robot control programs and software libraries has spawned several efforts to decouple the control programs as much as possible from the robots they are applied to. One line of abstraction is the use of robot description languages which provide models of a robot and then design and implement software components that work on the model components rather then the particular robot instance. The specialization to particular robots is then performed through the parametrization of the control programs.

A recent and successful example of such a robot description is the Unified Robot Description Format (URDF)[1], which can be used to specify the kinematics and dynamics, the visual representation and the collision model of a robot.

[1]http://www.ros.org/wiki/urdf

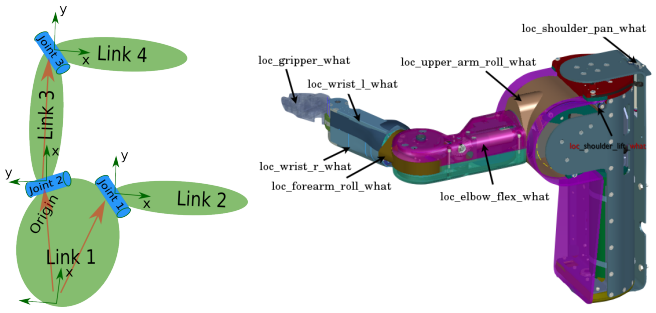

Fig. 1. Visualization of URDF's basic language elements: links and joints (left). Naming convention of labeling the parts of PR2 robot (right). Images by Willow Garage (http://www.ros.org/wiki), available under a Creative Commons Attribution 3.0 license.

By using the information of a robot description which is specified in URDF robot systems can calculate the 3D pose of a robot and detect potential collisions between a robot and its environment. Programmers can use the URDF description to visualize the respective robot and to simulate it in a physics-based simulator.

URDF specifies robot models using primarily two different language elements, namely *links* and *joints* (Fig. 1, left). A *link* element describes a rigid body part of a robot by specifying its origin, mass, inertia, geometry, visual appearance, and a collision model, whereas a *joint* element describes the connection of two *links*. The *joint* specification includes for example the joint type, e.g. revolute, continuous, or prismatic, and the joint limits. Fig. 2 shows the arm and hand of our robot TUM-Rosie and visualizes the corresponding URDF specification.

While URDF robot descriptions go a long way they are also still limited. In this paper we investigate one of these limitations: the lack of semantics of robot parts in the URDF

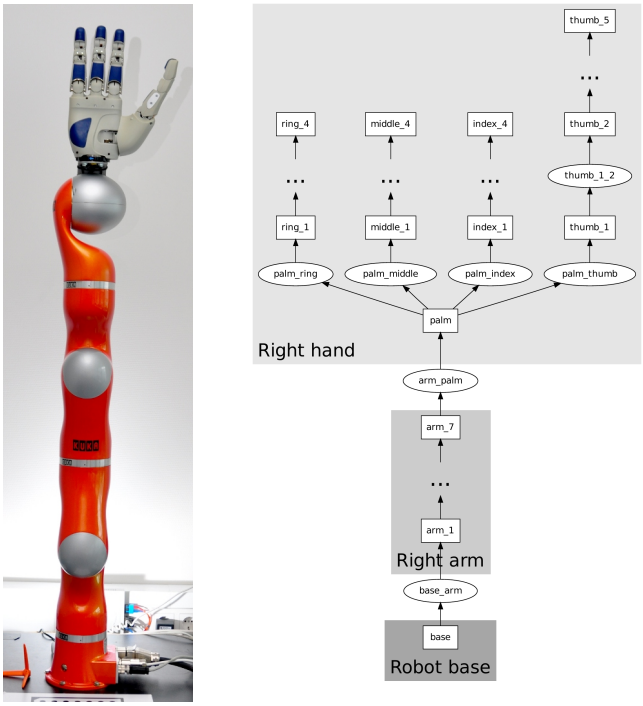

Fig. 2. TUM-Rosie's KUKA-lightweight LWR-4 arm with DLR-HIT hand (left), and visualization of the corresponding URDF specification (right). URDF's *links* and *joints* are visualized by boxes and ellipses respectively.

model. In URDF what a link corresponds to is only present in the link name. Thus, links have the name left-gripper-motor or left-gripper-encoder but the robot system does not know what this name means (Fig. 1, right).

In this paper, we explicitly *represent* the components of URDF descriptions in a symbolic knowledge base containing encyclopedic knowledge about robots and their components. We show how the robot specification format URDF can be embedded into a more powerful semantic robot description language, which we call SRDL. SRDL addresses not only robot descriptions but also the descriptions of actions and capabilities. It also provides inference mechanisms that allow reasoning about the ability of robots to perform certain actions.

SRDL enables robots to map an instruction like "Pick up the first chopstick with the middle finger and thumb", which can be retrieved from the World Wide Web, onto its own robot model and thereby generate it into an executable robot plan, as described in [1]. The action in this generated plan refers to the concepts *middle finger* and *thumb*, but not to the individual *links* or *joints*. Although *middle finger* and *thumb* are implicitly described by the URDF specification, the concepts cannot be located within the robot body.

The following dialog between a human and a robot illustrates the kind of questions that can be answered using SRDL:

HUMAN: Can you set the table with cups and plates?
ROBOT: Yes, I can.
HUMAN: How good are you in setting the table?
ROBOT: I've a success probability of 85%.
HUMAN: Can you set the table with silverware?
ROBOT: No, I cannot.
HUMAN: Why not?
ROBOT: I am missing an object model for perceiving silverware.

For answering these questions the robot basically has to match the action specifications and requirements with its own robot description and its collected experience.

To this end, this paper contributes the following:

- we develop a semantic description language for robots (including sensors, actuators, control algorithms and information objects);
- we extend the action representation established in [1] and link it to the developed robot descriptions; and
- we design and implement inference mechanisms for matching robots and actions and computing a success probability for actions based on experience.

The remainder of the paper is organized as follows: After discussing related work in Section II, we describe how the overall system is designed in Section III, i.e., how we represent robots and actions and how we link them via the concept of capabilities, and how we make inferences based on the represented knowledge. We present both, representations and inferences for sample problems in Section IV. Finally, we conclude in Section V.

II. RELATED WORK

Related work comprises three directions: robot descriptions, capability matchmaking and a combination of both.

A. Robot Descriptions

URDF was already discussed in some detail in Section I. Its application to kinematics, collision detection and visualization make it a powerful and indispensable tool for robot development. However, URDF is not designed for specifying robot components such as sensors, actuators, and control programs and matching those to action descriptions. In contrast, SRDL exactly addresses these issues and thereby aims at filling the gap between low- and high-level descriptions.

COLLADA[2] is an XML Schema designed for describing 3D objects including their kinematics. It mainly focuses on modeling information about scenes, geometry, physics, animations, and effects. But similar to URDF, it lacks elements for describing sensors, actuators and software.

Both [2] and [3] developed an OWL ontology to describe robots including their capabilities that operate in the domain of urban search and rescue. In contrast to SRDL, components and attributes are directly asserted to robots without modeling a kinematic chain. Also capabilities are directly asserted and not inferred from robot components.

The state-of-the-art of semantic sensor specifications is reviewed by [4]. An OWL ontology that focuses on the composition of sensors is developed in [5]. Efforts for providing a standard for sensor specifications are undertaken by the W3C Semantic Sensor Network Incubator Group[3]. As sensors are an important aspect within robotics, the work by [4], [5] and the Semantic Sensor Network Incubator Group is highly relevant for SRDL.

B. Capability Matchmaking

In general, the term *capability matchmaking* refers to the process of matching an advertisement of a capability with a request. In context of SRDL, a robot with its components corresponds to the advertisement whereas an action description resembles the request. The matching algorithm has to determine whether a robot is capable to perform an action.

In [6], agent and task descriptions are modeled in an ontology and matched by subsumption-based reasoning.

The LARKS system developed by [7] is built for matching web-based software agents. It employs several matching mechanisms based on semantic similarity and service specification structure to determine different degrees of matching.

OWL-S [8] allows to model the functionality and the process of web services semantically by using domain ontologies and predefined modeling constructs. Information about the functionality is used by the matchmaking algorithm to semantically compare advertisement and request.

As the approaches above, SRDL uses an ontology for representing knowledge. But a major difference lies in the way the capability advertisements and requests are matched.

[2] http://www.collada.org
[3] http://www.w3.org/2005/Incubator/ssn

Whereas the described systems match structurally equivalent specifications, SRDL matches action specifications to robots by verifying required capabilities and robot components.

C. Matching Sensor Descriptions to Tasks

In [9], the directions of robot descriptions and capability matchmaking are combined. The approach is able to compute compositions of assets that are jointly able to perform specific tasks in a military setting. Matching between tasks and assets is done by the means of capabilities.

When comparing SRDL to [9] it is noticeable that similarities exist. This is because we adopted and transferred some of their ideas to the domain of household robots. But there exist also considerable differences: the support of kinematics, sets of components providing a capability and modeling of diverse robot components like actuators, control programs and information objects. Furthermore, SRDL includes mechanisms to handle experiences about actions.

III. SYSTEM DESIGN

SRDL is used to specify robots, capabilities and everyday actions. Inference algorithms are used to match action descriptions to components of robots via the concept of capabilities. In the following, we first give an overview of the overall system and clarify the meaning of important concepts. Secondly, we present how robots, actions and capabilities are represented by the means of ontologies. And finally, we explain how the developed SRDL inference algorithms work.

A. System Overview and Terminology

The SRDL framework is tightly integrated within the knowledge processing system KNOWROB [10]. In terms of knowledge representation and reasoning this means that SRDL uses the Web Ontology Language OWL[4] for modeling knowledge about robots, capabilities and actions, and that it uses the logical programming language PROLOG for implementing inference algorithms.

Using an OWL representation allows SRDL to build on the inference mechanisms already available for OWL reasoning. For example, an object can be classified as being a *LaserScanner* based on the necessary and sufficient conditions for a *LaserScanner* defined in the OWL ontology. This classification can be used in SRDL inference if a *LaserScanner* is necessary for a certain action.

The integration within KNOWROB allows SRDL to build on an established knowledge base for robots. Whereas it mostly re-uses KNOWROB's knowledge about actions, SRDL extends the knowledge with information about robots, components and capabilities and relates it to existing knowledge. Furthermore, we established a separate module that automatically imports URDF specifications into SRDL's ontology.

But before we delve into the details of SRDL, it is important to understand the most relevant concepts within our approach, namely *Robot*, *Component*, *Action* and *Capability*. Definitions and examples for these concepts are given in Table I, whereas their relationship is visualized in Fig. 3.

[4] http://www.w3.org/TR/owl-features/

TABLE I
TERMINOLOGY

Concept	Definition	Examples
Robot	We define *Robot*, according to [11], as physical agent that performs actions by manipulating the physical world and that have effectors and sensors.	*TUM-Rosie*, *PR2*
Component	A *Component* of a robot is defined as hardware, e.g. sensors and effectors, software control program or information object, e.g. maps and object models.	*DLR-HIT-Hand*, *GraspPlanner*, *TeaCupModel*
Action	Following OpenCyc's definition of *PurposefulAction*, an *Action* is consciously, volitionally, and purposefully done by at least one actor.	*SetTheTable*, *PickingUpAnObject*
Capability	A *Capability* is the ability to perform a certain action. A robot has a *Capability* due to its components.	*PickAndPlaceCapability*

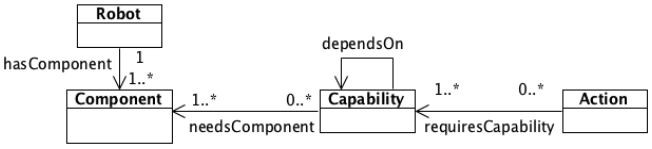

Fig. 3. Relationship between the concepts *Robot*, *Component*, *Capability* and *Action*

B. Ontological Representation

In this section we present how robot components, actions, and capabilities are modeled within SRDL using OWL.

1) Robot Components: A component is everything a robot consists of including but not limited to hardware components, software control programs and information objects.

Component Taxonomy: Each component is represented in the SRDL ontology as being an instance of a component class to capture its type of component and its specific properties. The component taxonomy is shown in Fig. 4 and defines component types, their relationships to other components and specific properties on component level. The component taxonomy covers hardware components (sensor and actuators), software components (control programs) and information objects (object models, data sets, and maps). The inheritance relationship of a taxonomy can be used to query

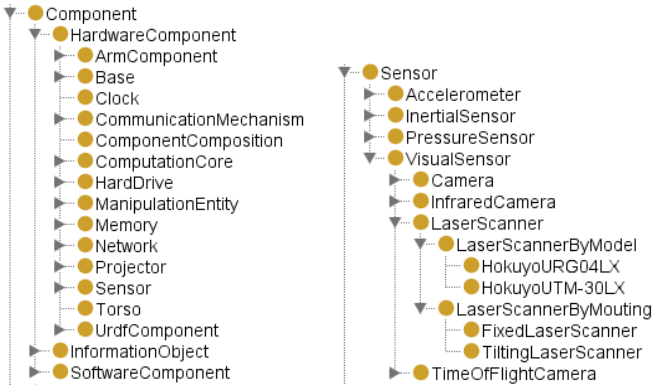

Fig. 4. (Part of) Component and sensor taxonomy

for certain types of components or components with specific properties, e.g. all sensors or all information objects.

Kinematic Chain and Component Composition: In order to capture the kinematic chain between hardware components of a robot the transitive object property *hasSuccessorInKinematicChain* can be used. Examples for such successor relationships in kinematic chain are a hand being the successor of arm or a joint being the successor of link. When using the URDF import module, an individual for each component is created and the successor relationships are asserted in this way. For example, in Fig. 2 each arrow corresponds to a *hasSuccessorInKinematicChain* assertion. Due to the transitivity of property *hasSuccessorInKinematicChain* it is easy to obtain all successors of an arbitrary component just by traversing the kinematic chain spanned by *hasSuccessorInKinematicChain* assertions.

The robot description at link and joint level is a rather low-level description. It can be used in order to calculate forward and inverse kinematics but it doesn't contain knowledge about the purpose of a certain link or joint. Often it is interesting to know what high-level component like hands or arms is formed by a set of links and joints. In order to be able to model such high-level components that consist of low-level components, a high-level component can be defined as a component composition, e.g. a *DLR-HIT-Hand*, and the set of low-level components that are part of the high-level component can be specified. This can be done using the object properties *hasBaseLinkInComposition* and *hasEndLinkInComposition*. As the kinematic chain is a tree structure there is always one base link of a component composition but there may be more than one end links (as is the case for a hand). For example, if we want to define the right hand pictured in Fig. 2 as a composite component, we would define *palm* as base link and *ring_4*, *middle_4*, *index_4*, and *thumb_5* as end links.

Asserting Membership of Component to Robot: So far we have discussed how single components and component compositions can be defined. In this paragraph we present how specific components can be asserted to be part of a specific robot. There are three ways how this can be achieved: first by the kinematic chain mechanism as explained above; second by directly asserting a component to a robot using the property *hasComponent*, and third by the property chain mechanism for software components.

2) Actions: Actions are events that are performed by a robot purposefully to change its environment and accomplish a goal.

Action Tree: An action consists of steps or sub-actions and those themselves are actions and can consist of sub-actions. This structure results in a hierarchical tree, called *action tree*. The action tree of an action is specified in SRDL using the object property *hasSubAction*. For each sub-action an assertion of property *hasSubAction* between the action and the sub-action is made. Fig. 6 shows an example for action *SetTheTable* consisting of three sub-actions and of one of the sub-actions' its sub-actions are shown.

Capability Dependencies: An action depends on a set of capabilities. The robot must possess all of these capabilities, otherwise it is judged to be incapable of performing the action. These capability dependencies of an action are specified using object property *requiresCapability*. For example, action *PickingUpAnObject* in Fig. 7 needs capability *MovingArmCapability* among others in order to perform the picking action.

Action Experience: A goal of SRDL is to enable a robot to use experience about past trials of an action in order to predict how likely it will be successful when performing an action. To accomplish this, experience information about an action has to be modeled. Datatype properties *hasNumSuccesses* and *hasNumTrials* are used to store the number of successful trials and total number of trials for an action.

3) Capabilities: Capabilities represent the ability of robots to perform certain actions.

Capabilities as Bridge between Robots and Actions: It is difficult to compare a robot and an action directly because they have very different characteristics: a robot has physical, spatial characteristics whereas an action abstract, hierarchical ones. In order to be able to match between robots and actions the concept of "capabilities" is introduced. An action depends on a set of capabilities and a set of components equips the robot with a certain capability. A capability is defined as a class in capability taxonomy and its properties are specified as explained below.

Component Dependencies: In order to exhibit a certain capability, a robot has to possess a set of components that cooperate and jointly enable the robot to exhibit the capability. For example, to exhibit the capability of navigating around in its environment, the robots needs components mobile base, a motion controller, a sensor, a object detection component and a map. Component dependencies of a capability can be modeled using object property *needsComponent*.

Sub-capability Dependencies: Additional to depending on components, a capability can also depend on other capabilities, so called "sub-capabilities". Sub-capabilities denote prerequisites of capabilities. This design allows modularization and reuse of definitions of capabilities and their component dependencies. The inference algorithm has to check sub-capabilities recursively when matching between robots and actions. Dependency of a capability on another capability can be modeled using object property *dependsOn*.

C. Inference Algorithms

The developed inference algorithms are realized by utilizing PROLOG's reasoning engine and additional libraries like the Semantic Web Library[5], but not employing of-the-shelf Description Logic reasoners. This allows us to easily extend the inference mechanism with other types of reasoning which are available in KNOWROB, e.g. commonsense reasoning. For instance, if a robot could not find cups for setting a table, it can propose to use glasses which are also of type container (cf. [12]).

[5]http://www.swi-prolog.org/pldoc/package/semweb.html

In this section we will discuss inference algorithms that match between robots and actions and calculate a success probability for performing an action. It is helpful to keep the knowledge model that is discussed above and summarized in Fig. 3 in mind when studying the inference algorithms.

1) Matching between Robots and Actions: The main question of SRDL is the matching between robots and actions: to judge if a robot can perform an action, totally independent of a real execution.

Feasibility of Action on Robot: The matching algorithm depends on the action tree and capability dependencies of actions. Robot *R* must fulfill two conditions in order to be able to perform an action *A*: First, all dependencies on capabilities of *A* must be fulfilled, meaning that all capabilities needed by *A* are available. Second, *R* must be able to perform all sub-actions of *A*.

In order to check those conditions, the inference algorithm iterates over all capability dependencies of *A* and checks for each single capability if it is available on *R* as described below. Further, the inference algorithm checks all sub-actions of *A* recursively. If a single missing capability or unfeasible sub-action of *A* is found, *A* is being judged to be unfeasible on *R*. If all capability dependencies are met and all sub-actions are feasible, *A* is judged to be feasible on *R*.

Verification of a Capability: There are two conditions a robot *R* must fulfill in order to exhibit capability *C*: First, all component dependencies of *C* have to be fulfilled. A single component dependency is fulfilled if *R* possesses an instance of the component specified by the component dependency. Second, all sub-capabilities of *C* must be available on *R* also.

The inference algorithm checks these two conditions for capability *C* by iterating over all component dependencies and sub-capabilities of *C* and checking each single one (in case of sub-capabilities recursively). If only one missing component dependency or unavailable sub-capability is found, *C* is judged to be unavailable on *R*.

Enumeration of Missing Capabilities and Components: In case of a negative matching outcome between a robot *R* and an action *A* it is desirable to know the reason for the failure. Thus, SRDL must be able to explain its inference result and enumerate all capabilities and components that are required for performing *A* but are missing on *R*.

The inference algorithm achieving this is an extension of the matching algorithm described above. It collects all capabilities that are required by *A* or one of its sub-actions, checks which of these capabilities are unavailable on *R* and produces a list of missing capabilities *MissingCapabilities*. Iteration over *MissingCapabilities* and collecting all their component dependencies results in the list of missing components. Note that sub-capabilities have to be considered recursively and a component dependency has to be checked if it is really nonexistent on *R* as missing capabilities usually depend on existing and nonexistent components.

2) Computing Success Probability of an Action: A second inference task of SRDL is to predict a success probability $P_{Success}(A)$ for robot *R* performing action *A*. We implemented two strategies for computing the success probability; both of them are based on the asserted numbers of trials and successes in the past.

Direct calculation: If numbers of successes and trials are asserted for action *A* the success probability $P_{Success}(A)$ can be calculated by dividing the number of successes by the number of trials: $P_{Success}(A) = \frac{NumSuccesses}{NumTrials}$.

Calculation via sub-actions: If none or only one number for trials and successes is asserted for action *A* on robot *R*, direct calculation is not possible. But the success probability of *A* can be calculated based on the success probabilities of the sub-actions of *A*. This is done by computing the success probabilities $P_{Success}(SubA)$ of all sub-actions of *A* and setting the success probability $P_{Success}(A)$ of *A* to the product of success probabilities of all sub-actions. The product of success probabilities of sub-actions is used assuming that all sub-actions are independent and due to the fact that the joint probability of independent events is the product of the probabilities of each single event. Sub-actions for which no success probability can be calculated are ignored and the calculation fails if there is no sub-action for which a success probability can be calculated.

IV. SAMPLE PROBLEMS

In this section we demonstrate the applicability of our approach by providing examples for robot and action descriptions and by explaining how the inference algorithms compute their consequences. We take a situation where our robot TUM-Rosie (Fig. 5) is supposed to set the table as sample problem. We first elaborate on the robot specification of TUM-Rosie, then present the action description for setting the table, and finally explain how the inference algorithms compute their results for various queries.

A. TUM-Rosie

The URDF description of TUM-Rosie can be automatically imported into the SRDL ontology. Thereby TUM-Rosie's low-level description including its kinematics is available in SRDL. Based on this low-level description, high-level components, i.e. sensors and actuators, are defined as instances of their respective classes. For example, the hand of TUM-Rosie is of type *DLR-HIT-Hand* and it is constituted of 18 instances of type *URDFLink* and 17 instances of type *URDFJoint* (Fig. 2). The information about TUM-Rosie's components can be retrieved from the ontology via a PROLOG interface. For example, the following query asks for all components of TUM-Rosie that are of type *Sensor*:

```
?- hasComponent(srdl:'TUM-Rosie', Comp),
   isType(Comp, srdl:'Sensor').

Comp = 'camera1-left' ;
Comp = 'camera2-right' ;
Comp = 'infraredCamera' ;
Comp = 'swissRangerTOF' ;
Comp = 'videreStereoOnChip' ;
Comp = 'hokuyo-shoulder' ;
Comp = 'hokuyo-rear' ;
Comp = 'hokuyo-front'.
```

Note, that *hasComponent* collects all components of TUM-Rosie by following the underlying kinematic chain

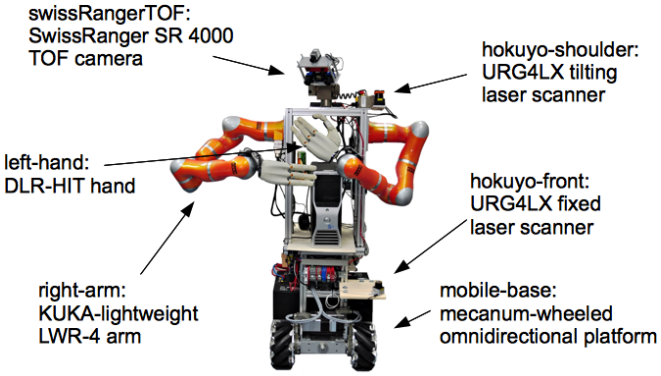

Fig. 5. TUM-Rosie and its high-level components

imported from URDF and that *isType* filters all instances whose type is derived from the concept *Sensor*.

Exemplarily, let us now look more closely at the last three instances, i.e. *hokuyo-<pos>*. All of them have are asserted to be of type *HokuyoURG4LX* and their membership of being of type *Sensor* is derived because of the subclass specifications of *Sensor*, *2D-VisualSensor* and *2D-LaserScanner* in the sensor taxonomy. Although all three instances have the same type, they differ in the way they are mounted on TUM-Rosie. But despite the different mounting locations, a more critical distinction is that the shoulder scanner can be tilted whereas the others are fixed. This distinction is reflected within the ontology where *hokuyo-front* and *hokuyo-rear* are additionally modeled as sub-concepts of *2D-FixedLaserScanner* whereas *hokuyo-shoulder* is additionally modeled as sub-concept of *3D-TiltingLaserScanner*. This differentiation within the taxonomy is highly relevant for TUM-Rosie's software components that rely on data gathered by these sensors. For example, perception routines for 3D object recognition require as input 3D point cloud data which can only be provided by an instance of type *3D-TiltingLaserScanner* but not *2D-FixedLaserScanner*.

After we highlighted some details about our robot TUM-Rosie, we now shortly consider the action description for setting a table and explain how these concepts are related to robot components via capabilities.

B. Setting the Table

For modeling actions we adopted the formalism established within KNOWROB. The action description for setting the table is represented by an instance of type *SetTheTable* which consists of several sub-actions. Each of them is a *PuttingSomethingSomewhere* action that refers to an object on which is acted on, e.g. an instance of type *Cup*. *PuttingSomethingSomewhere* has itself an ordered sequence of sub-actions: *PickingUpAnObject*, *CarryingWhileLocomoting* and *PuttingDownAnObject*. Again, these actions denote further sub-actions. A simplified excerpt of the hierarchical action representation of setting the table is depicted in Fig. 6.

We now explain how the action representations are related to robot components by examining the action *PickingUpAnObject* in detail. Picking up an object is basically an ordered sequence of actions, namely recognizing the object, moving the arm/hand towards it, grasping it, and finally lifting it. For doing so, the robot needs the respective capabilities. These capabilities are associated with *PickingUpAnObject* within the ontology. More specifically, the logical conjunction of role *requiresCapability* with range restrictions *3D-ObjectRecognitionCapability*, *MovingArmCapability*, and *GraspingCapability* form a super-class of *PickingUpAnObject*. The capabilities themselves refer to component classes they require, e.g. *3D-ObjectRecognitionCapability* depend on components of class *3D-VisualSensor* (e.g. *3D-TiltingLaserScanner*) and *3D-ObjectRecognitionAlgorithm*. The latter component has further dependencies, for recognizing a certain object the algorithm needs the respective *3D-ObjectModel*. Fig. 7 visualizes the relations between TUM-Rosie's components, its capabilities and the action of setting the table.

C. Inferences

To demonstrate the inference mechanisms of SRDL we examine the questions of the dialog presented in Section I.

Q1: Can you set the table with cups and plates?
For determining whether TUM-Rosie is able to set the table the inference algorithm matches the instance of *SetTheTableWithCupsAndPlates* with TUM-Rosie's robot description.

In the first step the algorithm collects all sub-actions of *SetTheTableWithCupsAndPlates*. The result is a list containing the following actions: *PuttingSomethingSomewhere-Cup*, *PuttingSomethingSomewhere-Plate*, *PickingUpAnObject*, *Reaching*, *TakingSomething*, *CarryingWhileLocomoting*, *PuttingDownAnObject*, *LoweringAnObject*, and *ReleasingGraspOfSomething*.

In the second step the algorithm proves for each action found in step one that TUM-Rosie is capable of performing it. For example, for proving that TUM-Rosie is capable of *PuttingSomethingSomewhere-Cup* all capability dependencies of this action sub-tree are collected and verified. As we saw earlier, *PickingUpAnObject* is one of the sub-actions in this action-tree. And since we cannot provide details for all sub-actions we give some explanations for *PickingUpAnObject*: *MovingArmCapability* and *GraspingCapability* are provided by TUM-Rosie's hardware components of type

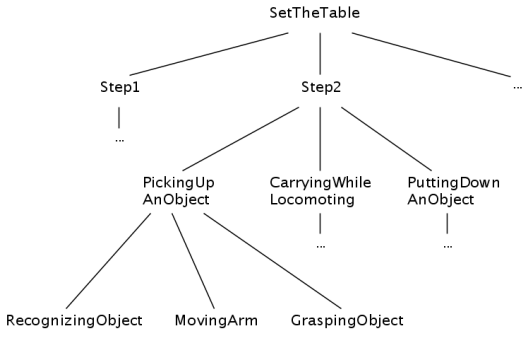

Fig. 6. Hierarchical action tree for setting the table.

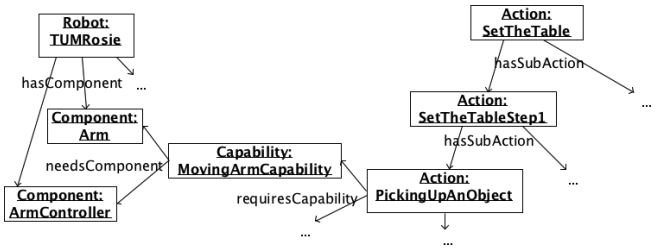

Fig. 7. Overview of how TUM-Rosie's components are related to capabilities and the action of setting the table.

KUKA-LWR4-Arm and *DLR-HIT-Hand* and their respective control programs. TUM-Rosie's component instances of type *3D-TiltingLaserScanner*, *3D-PerceptionPipeline*, and *3D-CupModel* provide the *3D-ObjectRecognitionCapability*. Similarly, the other actions retrieved in step one can be verified. Hence, the PROLOG predicate *matchRobotAndAction(srdl:'Rosie', srdl:'SetTheTableWithCupsAndPlates')* evaluates to true.

Q2: How good are you in setting the table?
The success probability for setting the table is calculated on the basis of TUM-Rosie's experience made in the past. If the the number of successful trials and the total number of trials are available for an action the success probability is directly given by the ratio. Otherwise the success probability of an action is the product of the success probabilities of all its sub-actions. In the case of *SetTheTableWithCupsAndPlates* the success probability, computed on made-up data, is 85%.

Q3: Can you set the table with silverware? Why not?
This prove is exactly the same as for *Q1*, despite that it fails. The reason why TUM-Rosie's description cannot be matched with *SetTheTableWithSilverware* is, that TUM-Rosie does not have an adequate object model. SRDL is designed that it can report why a prove fails by providing a list of missing capabilities and components. Here it says that the requirements for *3D-ObjectRecognitionCapability* are unmet because a component of type *3D-SilverwareModel* is missing.

V. CONCLUSIONS AND FUTURE WORKS

In this paper we investigate semantic robot description languages and inference mechanisms for matching robot descriptions with hierarchical action specifications. We present SRDL for describing robot components, capabilities and actions, and show that linking those descriptions allows to make inferences about the ability of robots to perform certain actions. The proposed system has been implemented and integrated within the knowledge processing system KNOWROB. We demonstrate the applicability of our approach by closely looking into the descriptions of our robot TUM-Rosie and the action specifications of setting the table, and by carefully following some sample inferences.

In future work, we aim to extend SRDL in several ways. We will integrate mechanisms that keep track of the current state of robot components, including sensors, actuators, control programs and information objects. Furthermore, we will account for constraints of components or sets of components. Our overall goal is to monitor future tasks and the current state of robot components needed by these tasks. For example, consider a robot that is supposed to use a rolling pin. Since it needs both hands for accomplishing this task, a problem can be detected when it is monitored that the robot already holds another object in one of its hands or one of its hand is broken. Such a monitor for tasks and components would help to observe potential problems, to recognize them in advance, and to deal with them proactively. Another direction we want to investigate is how SRDL could be connected more closely with robot planning and robot learning.

VI. ACKNOWLEDGMENTS

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems (CoTeSys)*, see also `www.cotesys.org`.

REFERENCES

[1] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web." in *IEEE International Conference on Robotics and Automation (ICRA).*, 2010.

[2] C. Schlenoff and E. Messina, "A Robot Ontology for Urban Search and Rescue," in *KRAS '05: Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems*. New York, NY, USA: ACM, 2005, pp. 27–34.

[3] R. Chatterjee and F. Matsuno, "Robot Description Ontology and Disaster Scene Description Ontology: Analysis of Necessity and Scope in Rescue Infrastructure Context," *Advanced Robotics*, vol. 19, no. 8, pp. 839–859, 2005.

[4] M. Compton, C. Henson, H. Neuhaus, L. Lefort, and A. Sheth, "A survey of the semantic specification of sensors," in *2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference,*, October 2009.

[5] M. Compton, H. Neuhaus, K. Taylor, and K. Tran, "Reasoning about sensors and compositions," *Proc. Semantic Sensor Networks*, p. 33, 2009.

[6] Y. Gil and S. Ramachandran, "Phosphorus: a task-based agent matchmaker," in *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA: ACM, 2001, pp. 110–111.

[7] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace," *Autonomous agents and multi-agent systems*, vol. 5, no. 2, pp. 173–203, 2002.

[8] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. Mcguinness, E. Sirin, and N. Srinivasan, "Bringing semantics to web services with owl-s," *World Wide Web*, vol. 10, no. 3, pp. 243–277, 2007.

[9] A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, G. Pearson, T. Pham, and T. Porta, "Matching sensors to missions using a knowledge-based approach," *SPIE Defense Transformation and Net-Centric Systems*, vol. 2008, 2008.

[10] M. Tenorth and M. Beetz, "KnowRob — Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2009.

[11] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Prentice hall, 2009.

[12] C. Galindo, J. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008.