Project Acronym:	ACAT
Project Type:	STREP
Project Title:	Learning and Execution of Action Categories
Contract Number:	600578
Starting Date:	01-03-2013
Ending Date:	30-04-2016

Deliverable Number:	D4.2
Deliverable Title:	Structured reactive controllers and error recovery
Type (Internal, Restricted, Public):	PU
Authors :	Gheorghe Lisca, Daniel Nyga, Michael Beetz, Andrej Gams and Aleš Ude
Contributing Partners:	UoB, JSI

Contractual Date of Delivery to the EC:	31-05-2015
Actual Date of Delivery to the EC:	01-06-2015

Contents

1 Executive Summary

Generally, WP4 of the ACAT project focuses on developing planning and execution mechanisms. In particular, three of the main objectives of this work package are:

- designing and implementing the plan schemata,
- creating structured reactive controllers for ACAT sequences,
- developing movement descriptions.

All three support the bigger objective of extending the execution engine. We followed these objectives in the context of a robotic agent which is challenged to perform chemical experiments in a chemistry laboratory and industrial assembly tasks.

2 Structured reactive controllers in CHEMLAB scenario

Top-down, the robotic agent is provided with lab protocols which describe in natural language the chemical experiment to be performed. Using the PRAC system [R6], the robot understands each protocol's instruction and infers which actions must be performed and which ADTs must be queried (by the plans at execution time) for manipulation particularities. Associated with each inferred action we developed multiple plan schemata. At analysis time PRAC's probabilistic inference runs on each instruction and its results fully parametrize one of the plan schemata associated to the inferred action. Within the context of CHEMLAB scenario we developed plan schemata for actions like:

- opening / closing containers by unscrewing / screwing their caps
- loading / unloading test tubes into centrifuge

Based on a fully parametrized plan schema the robot is able to identify the most competent structured reactive controller (plan) to be run in order to perform the manipulations implied by each given action. At execution time the inferred plan will query the inferred ADT for the missing manipulation details. If some execution details are missing then the plan will compute them and update them back into the queried ADT. For example if the unscrewing plan queries the unscrewing ADT for which trajectory to execute and the unscrewing ADT doesn't contain this trajectory then the plan will compute it and update it into the unscrewing ADT.

The collection of all structured reactive controllers is stored into a library (plan library). It contains action specific plans and action abstract plans. The action specific plans build on top of action abstract plans. The latter ones are controlling the final robotic motions.

The (currently under review) paper [A1], which is attached to this deliverable, explains in more details the concepts introduced above.

3 Structured reactive controllers in IASSES scenario

In IASSES scenario dynamic movement primitives (DMPs) are used for robot control. They can be directly extracted from action data tables (ADTs) and utilized to encode robot skills, which can be represented either in Cartesian or in joint space. In their basic form, DMPs consist of a linear second order attractor system (system of differential equations) with the added nonlinear forcing term, which is applied to adapt the simple second order attractor dynamics to the specific robot skill. DMPs have many favourable properties, e. g. they contain open parameters that can be used for learning without affecting the overall convergence and stability of the system, They can control timing without requiring an explicit time representation, they are robust against perturbations and they can be modulated to adapt to different requirements. This last property make them suitable as reactive controllers, which is the main topic of this section.

For clarity and easier reading we provide the basic equations of the DMP in the following text as proposed in [R4]. For a single degree of freedom (DOF) denoted by y, in our case one of the external task-space coordinates, a DMP is defined by the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \quad (1)$$

$$\tau \dot{y} = z. \quad (2)$$

$f(x)$ is defined as a linear combination of nonlinear radial basis functions

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \quad (3)$$

$$\Psi_i(x) = \exp\left(-h_i \left(x - c_i\right)^2\right), \quad (4)$$

where c_i are the centers of radial basis functions distributed along the trajectory and $h_i > 0$ their widths. Provided that parameters $\alpha_z,\ \beta_z,\ \tau > 0$ and $\alpha_z = 4\beta_z$, the linear part of the system (1) – (2) is critically damped and has a unique attractor point at $y = g,\ z = 0$. A phase variable x is used in (1), (3) and (4). It is utilized to avoid direct dependency of f on time. Its dynamics is defined by

$$\tau \dot{x} = -\alpha_x x, \quad (5)$$

with initial value $x(0) = 1$. α_x is a positive constant.

The weight vector $\mathbf{w}$, composed of weights w_i, defines the shape of the encoded trajectory. Ijepeert et al. [R4] describe the learning of the weight vector. Multiple DOFs are realized by maintaining separate sets of (1) – (4), while a single canonical system given by (5) is used to synchronize them. For movements defined in task space, we proposed an extension described in the deliverable **D4.1** and the paper attached to this deliverable [R7].

3.1 Dynamic movement primitives as structured reactive controllers

DMPs can be used as reactive controllers because they have several favourable properties and can adapt to external signals. Adaptations can be either temporal or spatial. In the following we first provide a list of adaptation to external signals.

Robustness to perturbations

The dynamic equations of DMPs can be physically interpreted as a spring-damper system of point-mass, where the nonlinear part represents the accelerations that ensure the tracking of this critically damped mass of the desired trajectory. The accelerations are often also termed "the forcing term". In the case of a perturbation, the dynamic system will smoothly approach the original desired trajectory, with the approach velocity depending on the parameters α_z, β_z of the DMP.

Robustness to goal change

DMPs also provide a facility to change the final configuration, i. e. the goal of the movement. Similarly as in case of perturbations, the DMP dynamic system gracefully adapts the desired trajectory to the new goal of the movement. In ACAT this property has been demonstrated for Cartesian space DMPs [R7].

External limit modulation

One of possible spatial modulations includes a simple repulsive force to avoid moving beyond a given position, either in the task space or in the joint space, depending where the DMP is defined. Such a repulsive force can be specified by modifying (2) into

$$\tau \dot{y} = z + h(y), \tag{6}$$

while leaving (1) in the original form. A simple repulsive force to avoid hitting y_L can be defined as [R3]

$$h(y) = -\frac{1}{\gamma(y_L - y)^3}, \tag{7}$$

where y_L is the known limit.

Obstacle avoidance in DMPs

Obstacle avoidance can very easily be implemented with the introduction of an additional forcing term, which pushes the trajectory away from the obstacle. In the following we explain how to achieve obstacle avoidance for a 3-dimensional DMP trajectory in task space.

The 3-D position vector of the 3 DOF discrete dynamical system is encoded by $\mathbf{p} = [x, y, z]^T$. The objective is to generate a reaching movement to a goal state $\mathbf{g} = [g_1, g_2, g_3]^T$. On the way to the goal state, an obstacle is positioned at $\mathbf{o} = [o_1, o_2, o_3]^T$ and needs to be avoided. A suitable coupling term $\mathbf{C}_t = [C_{t,1}, C_{t,2}, C_{t,3}]^T$ for the obstacle avoidance can be formulated as follows:

$$\mathbf{C}_t = \gamma \, \mathrm{sig}\left(\|\mathbf{o} - \mathbf{p}\|\right) \mathbf{R}\dot{\mathbf{p}} \; (\pi - \phi) \, \exp\left(-\beta\phi\right), \tag{8}$$

where

$$\phi = \arccos\left(\frac{(\mathbf{o} - \mathbf{p})^T \dot{\mathbf{p}}}{\|\mathbf{o} - \mathbf{p}\| \|\dot{\mathbf{p}}\|}\right), \tag{9}$$

$$\mathrm{sig}(x) = \frac{1}{1 + e^{\eta(x-d)}}, \tag{10}$$

$$\mathbf{R} = \exp\left(\left(\frac{\pi}{2} - \phi\right)\mathbf{n}\right), \tag{11}$$

$$\mathbf{n} = \frac{(\mathbf{o}-\mathbf{p})\times\dot{\mathbf{p}}}{\|\mathbf{o}-\mathbf{p}\|\|\dot{\mathbf{p}}\|}. \quad (12)$$

γ, β, and η are the scaling factors and d is the distance at which the obstacle should start affecting the robot's motion. The coupling term as defined above generates a velocity component that is in a plane defined by the vectors $\mathbf{o}-\mathbf{p}$ and $\dot{\mathbf{p}}$. It is also orthogonal to the line $\mathbf{o}-\mathbf{p}$, which is connecting the tip of the robot and the obstacle.

We can ensure that the tip of the robot, i.e., the end-effector, avoids the obstacle by adding this coupling term $\mathbf{C}_t$, or the new forcing term, to Eq. (1)

$$\tau\dot{\mathbf{z}} = \alpha_z(\beta_z(\mathbf{g}-\mathbf{p})-\mathbf{z}) + \mathbf{f}(x) + \mathbf{C}_t \quad (13)$$

Slow-down feedback for error recovery

As the DMPs are not explicitly dependent on time, but on the phase of the motion, the evolution of the phase can be used to accelerate or slow-down the motion, resulting in a temporal modulation. In the following we present the slow-down feedback.

In ACAT we implemented DMP phase stopping in Cartesian space [R7] and applied it to realize force-based manipulation skills [R1]. For this purpose, the original equation for phase (5) was replaced with

$$\tau\dot{x} = -\frac{\alpha_x x}{1+\alpha_{px}\left(\|\tilde{\mathbf{p}}-\mathbf{p}\| + \gamma\,\mathrm{d}(\tilde{\mathbf{q}},\mathbf{q})\right)}, \quad (14)$$

where $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ are the actual position and orientation (represented as unit quaternion) of the tool center point, respectively, and $\mathbf{p}$ and $\mathbf{q}$ the corresponding DMP control outputs.

Note that $\|\tilde{\mathbf{p}}-\mathbf{p}\| + \gamma\,\mathrm{d}(\tilde{\mathbf{q}},\mathbf{q})$ is the trajectory tracking error. In the case of large tracking errors, the error value $\|\tilde{\mathbf{p}}-\mathbf{p}\| + \gamma\,\mathrm{d}(\tilde{\mathbf{q}},\mathbf{q})$ becomes large which in turn makes the phase change $\dot{x}$ small.

Thus the phase evolution is stopped until the robot reduces the tracking error and starts following the trajectory again.

To recover from errors when executing force-based manipulation skills, we monitor the differences between the desired forces $\mathbf{F}_d$ and torques $\mathbf{M}_d$ and the currently measured values $\mathbf{F}$ and $\mathbf{M}$, respectively

$$\mathbf{e}_p = \mathbf{q} * (\mathbf{F}_d - \mathbf{F}) * \overline{\mathbf{q}}, \quad (15)$$
$$\mathbf{e}_q = \mathbf{q} * (\mathbf{M}_d - \mathbf{M}) * \overline{\mathbf{q}}, \quad (16)$$

where $\mathbf{q}$ is the current orientation of the tool center point. In this case the phase stopping criterion (14) can be replaced with

$$\tau\dot{x} = -\frac{\alpha_x x}{1+\alpha_{px}\left(\|\mathbf{e}_p\| + \gamma\,\|\mathbf{e}_q\|\right)}. \quad (17)$$

and the phase evolution is stopped until the robot has reduced the force-torque error.

DMP sequencing for execution of structured plans

In IASSES scenario we use the concept of semantic event chains [R2] to generate sequences of action chunks as specified by ADTs. At the control level, it is necessary to smoothly chain these action chunks. Smooth transitions between consecutive DMPs can be achieved by using mechanisms provided in Nemec and Ude [R5].

4 Conclusions

In this deliverable we described mechanisms to implement structured reactive controllers and error recovery in CHEMLAB and IASSES scenario. In CHEMLAB scenario we developed two main types of structured reactive controllers: action specific and action abstract structured reactive controllers. On top of action specific structured reactive controllers we develop the plan schemata which are parametrized with the results of PRAC's probabilistic inference. Under action abstract structured reactive controllers we are able to execute DMPs or different trajectories stored in ACAT's ADTs.

In IASSES scenario we use DMPs to execute complex plans. DMPs provide many mechanisms that make them suitable as structured reactive controllers. In ACAT we extended these mechanisms to dynamic movement primitives in Cartesian space and showed how DMP phase stopping can be used for error recovery in force-based manipulation skills.

References

[R1] F. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude. "Adaptation of manipulation skills in physical contact with the environment to reference force profiles". In: *Autonomous Robots* (2015). DOI: 10.1007/s10514-015-9435-2.

[R2] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. "Learning the semantics of object-action relations by observation". In: *The International Journal of Robotics Research* 30.10 (2011), pp. 1229–1249.

[R3] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič. "On-line learning and modulation of periodic movements with nonlinear dynamical systems". In: *Autonomous Robots* 27.1 (2009), pp. 3–23.

[R4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. "Dynamical movement primitives: Learning attractor models for motor behaviors". In: *Neural Computations* 25.2 (2013), pp. 328–373.

[R5] B. Nemec and A. Ude. "Action sequencing using dynamic movement primitives". In: *Robotica* 30.5 (2012), pp. 837–846.

[R6] D. Nyga and M. Beetz. "Everything robots always wanted to know about housework (but were afraid to ask)". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, 2012, pp. 243–250.

[R7] A. Ude, B. Nemec, T. Petrič, and J. Morimoto. "Orientation in Cartesian Space Dynamic Movement Primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, 2014, pp. 2997–3004.

Attached papers

[A1] G. Lisca, D. Nyga, F. Bálint-Benczédi, H. Langer, and M. Beetz. "The Chemist Robot Extracting DNA". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, (under review).

The Chemist Robot Extracting DNA

Gheorghe Lisca, Daniel Nyga, Ferenc Bálint-Benczédi, Hagen Langer and Michael Beetz

***Abstract*— Autonomous mobile robots are employed to perform increasingly complex tasks which require appropriate task descriptions, accurate object recognition, and dexterous object manipulation. In this paper we will address three key questions: *How to obtain appropriate task descriptions from natural language (NL) instructions, how to choose the control program to perform a task description, and how to recognize and manipulate the objects referred by a task description?* We describe an evaluated robotic agent which takes a natural language instruction stating a step of DNA extraction procedure as a starting point. The system is able to transform the textual instruction into an abstract symbolic plan representation. It can reason about the representation and answer queries about what, how, and why it is done. The robot selects the most appropriate control programs and robustly coordinates all manipulations required by the task description. The execution is based on a perception sub-system which is able to locate and recognize the objects and instruments needed in the DNA extraction procedure.**

I. Introduction

As the area of autonomous robot manipulation gets more mature it is also getting more important that we better understand the nature of the underlying information processing mechanism by building complete systems that perform human-scale manipulation tasks. The importance of research concerning the building of complete robotic agents can not be overestimated. We have made impressive progress in component technologies such as navigation, grasping, and perception but so far it is not clear how the individual components have to be pieced together to produce competent autonomous activity.

Consider, for example, the control of robot motions. We see many systems that produce and often even learn to produce very sophisticated motion patterns such as flipping a pancake or catching a ball in a cup. However, these systems have no idea of what they are doing. You cannot ask them about the desired and undesired effects of actions, how the course of action could be changed in order to avoid some unwanted side effect, and so on. For example, the result of pouring a chemical substance into a container might cause an explosion.

The reason for this situation is that in order to learn or generate sophisticated motions you have to completely formulate the problem in a mathematical model that is then solved in order to generate a control law that constitutes a desirable mathematical solution. The problems of how the mathematical models and computational problems can be generated by a robot tasked with a NL instruction and

Institute for Artificial Intelligence, Department for Computer Science, University of Bremen, Germany. {lisca, nyga, balintbe, hlanger, beetz}@cs.uni-bremen.de

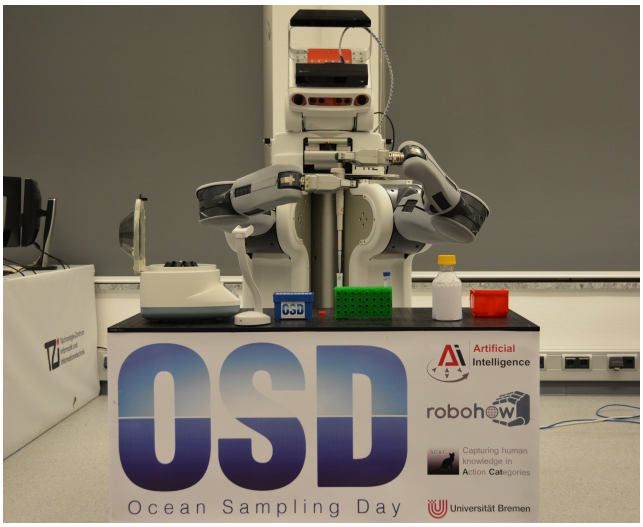

Fig. 1: Uni-Bremen's PR2 pipetting.

looking at a particular scene has not received sufficient attention. The same holds for the problem of enabling robots to answer questions about what they are doing, how, why, what could possibly happen, and so on.

In this paper we describe a robotic agent that is capable of autonomously conducting chemical experiments with ordinary laboratory equipment based on NL instructions for these experiments. The actions that the robotic agent is to perform include taking tubes, opening and closing them, putting them into a rack, mixing chemical substances through pipetting, and operating a centrifuge by opening and closing it, loading and unloading it, and pushing the start button.

The application is interesting because it requires the robot to perform only a small set of manipulation actions but by combining these actions in different ways and performing them with different substances and quantities the robot can potentially perform thousands of different chemical experiments by reading and executing instructions for experiments. In addition, large knowledge bases about chemistry that are available in standardized and machine readable form in the semantic web enables us to realize knowledgeable robots with comparatively little effort.

The main contribution of this paper is the realization of a complete robot agent that can autonomously conduct (carefully selected) chemical experiments. In this context the main technical contributions are:

1) The generation of abstractly parameterized plans from NL instructions, which means that a language instruction such as "neutralize 250ml hydrochloric acid" is translated into an abstractly parameterized action de-

Preprint submitted to 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 4, 2015.

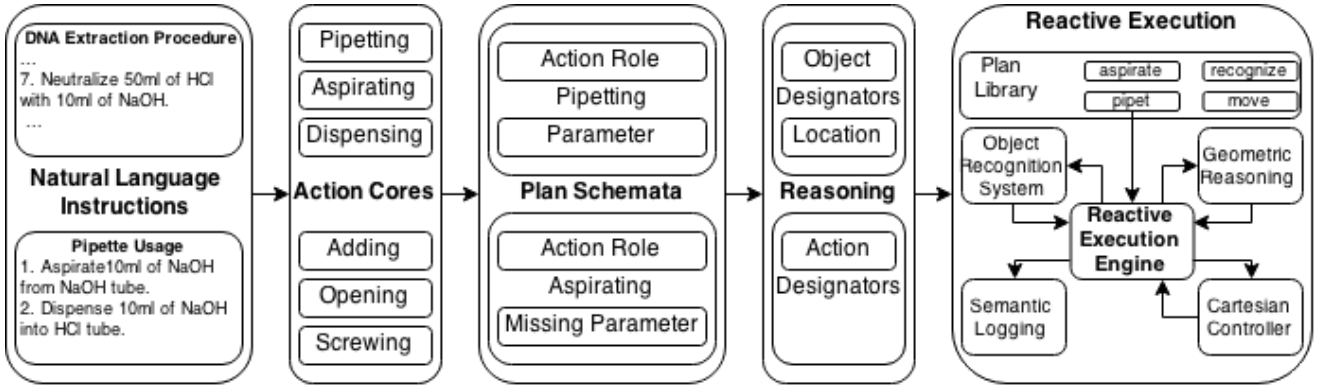

Fig. 2: Agent's Conceptual Architecture

scription

```
(perform
  (an (action
        (type pipetting)
        (object-acted-on ...)
        ... ))
```

which names the plan to be called, namely pipetting, and assigns each of the formal parameters of the pipetting plan an abstract symbolic parameter description. To deal with the incompleteness and ambiguities of NL instructions the robotic agent employs first-order probabilistic reasoning to carry out this interpretation step.

2) A knowledge-enabled perception component that is able to recognize symbolically described objects such as "the pipette containing the acid substance" or "the lid of the tube in the rack" and localize them accurately enough to allow for high precision manipulation tasks such as putting a tip on the pipette.
3) The perceptually grounded execution of abstractly parameterized plans that takes abstract descriptions of objects, locations, and actions and translates them into specific numeric parameters such as the 6D pose of the pipette for releasing the content of the pipette.
4) The acquisition and the reasoning about episodic memories of chemical experiment activities that enable the robotic agent to answer queries about what it did in the episode, how, why, what happened, etc.

The robotic agent was shown in a public demonstration (see youtube video[1]), in which it participated in the Ocean Sampling Day.

The remainder of this paper is structured in the following way: Section II will present an overview of our system. How does NL understanding happen will be detailed in Section III. Section V will explain the first symbolic representation of an instruction. In Section V-A the reasoning mechanism which separates the symbolic task descriptions in more specific symbolic descriptions for action, object and location, will be presented. How are the specific descriptions used at runtime will be explained in Section V-B. The experiments and drawn conclusions will be summarized in the sections VI respectively VIII.

[1] https://www.youtube.com/watch?v=vBZ-Vm5nvBs

II. Conceptual Architecture of the Robotic Agent

From describing the *DNA Extraction Procedure* and *Pipette Usage* through NL instructions to having the robot reactively pipetting: *which are the key steps an intelligent robot has to go trough in order to parametrize its control programs from NL?* Figure 2. The first step is to understand the NL instructions which task him.

The two sets of NL instructions for neutralization and pipette usage are parsed using Stanford parser [1] and for each instruction word the identified syntactic role is stored into a probabilistic first order relational database. Using *wordnet* [2] for each word it's meaning is identified. Based on the meanings and syntactic roles of instruction's words, the action cores for *pipetting*, *aspirating* and *dispensing* which match the best given instructions are identified. The matching process tries to associate action roles to instruction's words. The roles of action core's which don't have an instruction word associated with them, will be used to infer instruction's implicit words which are missing from instruction's text. *aspirating* and *dispensing* involve the instrument *pipette* which doesn't explicitly appear in *Pipette Usage* instructions' text.

Each action core has a *Plan Schema*, detailed in Section V, associated with it. A plan schema groups into a tuple the action verb and the action roles from the same action core. The tuple can be regarded as an abstract description of an action. Defined in this way, a plan schema is fully parameterizable by its associated action core. A fully parametrized plan schema is a plan schema for which all its action roles were replace by instruction's specific entities.

In the first phase of the third step, from previously obtained fully parametrized plan schema, the *Reasoning Mechanism*, detailed in Section V-A, extracts the symbolic descriptions of objects, locations and actions. We call these symbolic descriptions: *designators*. In the second phase of this step, from the freshly extracted action designator, the

Preprint submitted to 2015 IEEE/RSJ International Conference
on Intelligent Robots and Systems. Received March 4, 2015.

reasoning mechanism infers which control program is the most competent for performing the manipulations required by the action description. We call the control program simply *plan* and the entire collection of control programs *Plan Library*.

In the fourth step, from the plan library, the *Reactive Execution Engine*, detailed in Section V-B, retrieves the plan inferred by the reasoning mechanism. The plan gets the previously extracted object and location designators as parameters and runs as a normal program. At plan's runtime the reactive execution engine triggers the *Semantic Logging* [3] module to log plan's context, goals events and the sensor data which influenced robot's decisions. *OpenEASE* [4] is the web-based knowledge service which collects the data about robot's runtime experiences and makes it available to other robots.

III. Generating Abstractly Parameterized Plans from NL Instructions

Robotic agents acting in human environments must be capable of proficiently performing complete jobs in open environments that they have not been preprogrammed for. A promising direction towards this skill, which has gained a lot of attraction in the recent couple of years, is to equip robots with the capability to acquire new high-level skills from interpreting NL instructions, which can be found in abundance on the web. Instruction sheets provide a rough and sketchy sequence of actions that needs to be executed in order to accomplish a task.

However, these instructions typically are written by humans and are intended for human use, so they lack massive amounts of information about how particular action steps are to be executed, on which objects they are to be performed, which utensils to be used and so on. In addition, a specific action can be achieved in different ways or even must be achieved in a very particular way, depending on the current context the action takes place in. As an example, consider an action like 'add hydrochloric acid', which might be taken from an instruction sheet describing a chemical experiment. It is neither specified explicitly where to add the acid to, how much of it, or how to add it. If the amount that is to be transferred is very small and accurately specified, such as '5 drops', one may want to choose a pipette for doing the addition. Conversely, if 100 ml should be transferred, one should use a measuring cup and pour directly from the container where the acid is located.

Thus, instructions stated in NL are severely vaguely formulated, they are ambiguous and underspecified, and proficiently performing instructions requires a robotic agent to *interpret what is meant* by an instruction by *understanding what is given* and *inferring what is necessary*.

Probabilistic Action Cores (PRAC) [5] are action-specific first-order probabilistic knowledge bases that are able to interpret instructions formulated in NL and infer the most probable *completion* of an action with respect to its abstract, symbolic parameterizations. More specifically, action cores can be regarded as abstract patterns of actions and events,

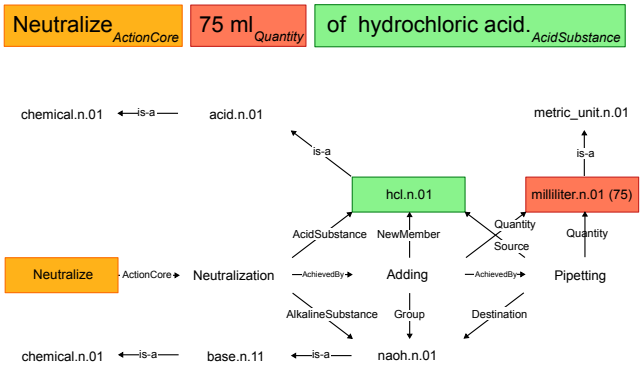

Fig. 4: Exemplary instance of action cores and their action roles for the 'neutralization' example. The colored nodes are given as evidence, whereas the gray nodes and the role assignments need to be inferred.

which have a set of formal parameters attached that must all be known in order to parameterize a robot plan appropriately. As an example, consider the NL instruction "neutralize 10 ml of hydrochloric acid." In this example, a 'Neutralization' (in a chemical sense) represents an action core, which has attached to it two *action roles*, namely an *AcidSubstance* and an *AlkalineSubstance*, which both must be known in order to perform the neutralization. However, in the original instruction, the alkali counterpart is not specified. From a probabilistic point of view, one can query for the *most likely role assignment* given what is explicitly stated in the instruction:

$$\arg\max_{c} P\left(\textit{is-a}(s,c) \;\middle|\; \begin{array}{c} \textit{action-core}(a, \textit{Neutralization}) \\ \textit{AcidSubstance}(a, \textit{hcl}) \\ \textit{is-a(hcl, hcl.n.01)} \\ \textit{AlkalineSubstance(a, s)} \end{array} \right),$$

i.e. we are querying for the most probable type c of an entity s that fills the *AlkalineSubstance* role, given the action core *Neutralization* and the type *hcl.n.01* of the *AcidSubstance* role. A graphical representation of this action core is given in Figure 4.

In many cases, it is not sufficient to consider the action verb as it is stated in an instruction. In our example, the neutralization is not a directly executable action. It rather denotes a chemical process that needs to be triggered. A robot thus needs to be equipped with reasoning capabilities that allow to infer *how* a particular action can be achieved. The neutralization, for instance, can be achieved by *adding* the alkaline substance to the acid that is to be neutralized, and, since the amount of 10 ml is small an accurately specified, the adding action can be achieved by pipetting one substance to the other. PRAC uses a dedicated action role *AchievedBy*, which enables to reason about which action can be achieved by some other action, given its abstract parameterization.

PRACs are implemented as Markov logic networks [6], a powerful knowledge representation formalism combining first-order logic and probability theory. A key concept in PRAC is heavy exploitation of taxonomic knowledge, which enables to learn PRACs from very sparse data. By exploiting

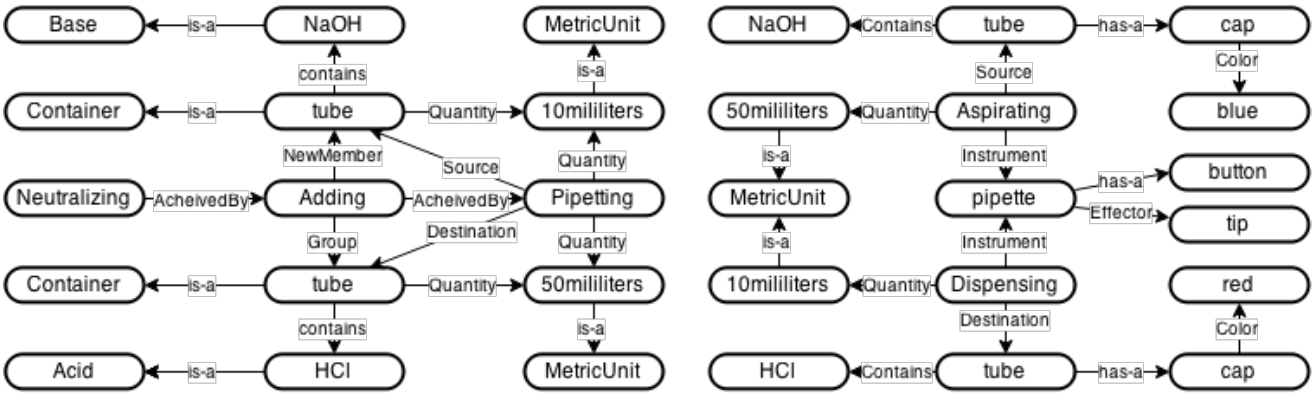

Fig. 3: Action Cores: Neutralizing, Pipetting, Aspirating and Dispensing

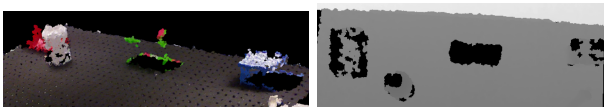

Fig. 5: Sensory input for the demonstration scenario

the relational structure of concepts in a real-world taxonomy like the WordNet lexical database, PRAC can perform reasoning about concepts that it has not been trained with.

Action cores can be regarded as conceptualizations of actions that can have an abstract plan schemata attached to them. In these cases, action roles interface the formal parameters of the plan.

IV. Knowledge-enabled Perception of Experiment Setups

Detecting the necessary objects for executing the experiment becomes challenging, given the nature of these, the tasks that need to executed and the noisy input data (Figure 5). Furthermore it is not enough to detect the labels of each object, but identifying parts of them is also necessary (e.g. opening of a bottle or a tube). To address these challenges we use a knowledge-driven approach, where the perception system can reason about the objects it perceives and infer the correct processing step for detecting the parts of the objects need to be manipulated[7].

This is done through a two step process. First the objects, their corresponding class lables , visual properties and their initial pose are detected. Since the objects are represented in our knowledge-base, based on their class labels we have access to information that can help further examining them. In Figure 6 for example the objects *rack* and *bottle* have the property *contains*, from which we can infer the next processing step necessary to find the openings of the bottle or detect if the tubes are closed or open.

We define Prolog rules which are able to deduce parameterizations for more general perception algorithms, in order to detect the necessary parts of the objects. For example:

```
fitCircle(Object, Radius) :-
  category(Object, 'contaier'),
  object-part(Object, Opening),
```

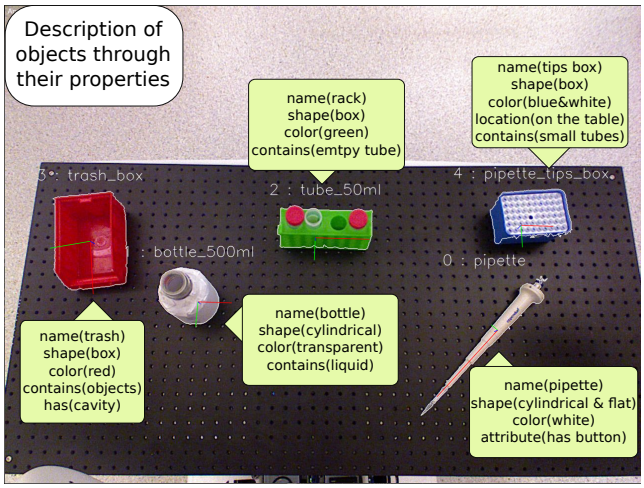

Fig. 6: Description of the perceived objects

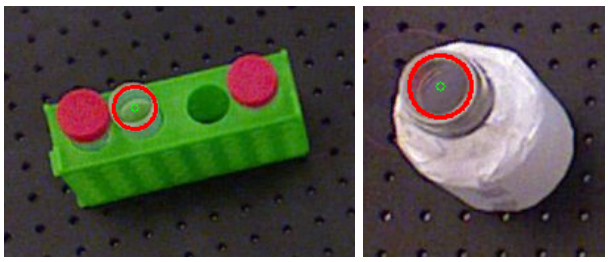

Fig. 7: Circle fitting to find the opening of the containers

```
geom_primitive(Object, 'cylindrical'),
radius(Opening, Radius).
```

deduces the radius of a circle that needs to be fit to an object that is a cylindrical container. The results of the perception system after executing this query are shown in Figure 7.

V. Perceptually Grounded Execution of Abstractly Parametrized Plans

As we introduced it in Section II, a plan schema is a template defined over an action verb and the set of action roles defined within an action core.

$$(\langle Action\ Verb \rangle\ (\langle Action\ Role_0 \rangle\ \ldots \langle Action\ Role_n \rangle))$$

Preprint submitted to 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 4, 2015.

Code Excerpt 1 Plan Schemata

```
(pipetting
  (from ((source Pipetting)
         (chemical (contains (source Pipetting)))
         (type (is-a (source Pipetting)))))
  (into ((destination Pipetting)
         (chemical (contains (destination Pipetting)))
         (type (is-a (destination Pipetting))))))

(aspirating
  (from ((source Aspirating)
         (chemical (contains (source Aspirating)))))
  (amount (quantity Aspirating))
  (into ((instrument Aspirating))))

(screwing
  (the (mobile-object Screwing))
  (on (fixed-object Screwing))
  (using (tool Screwing)))
```

Code Excerpt 2 Action Designators

```
(pipetting
  (from Container)
  (with Instrument)
  (into Container))

(aspirating
  (from Container)
  (amount Quantity)
  (into Instrument))
```

A fully parameterized plan schema guides the reasoning mechanism in inferring the most adequate plan which has to run in order for robot to execute the instructions with which is tasked. *pipetting* plan schema, Code Excerpt 1, states that the pipetting action firstly needs a *source* which *contains* a specific chemical and is of *type* container and secondly it needs a *destination* which *contains* another specific chemical and is of *type* container too. *pipetting* plan schema starts to capture *what* has to be done for the *pipetting* action. Once the pipetting action schema is fully parameterized it specifies exactly with which objects the pipetting action has to be performed. *pipetting* fully parameterized plan schema doesn't contain *how* pipetting has to be done. *How* an action will be done only the control programs know. *screwing* fully parameterized plan schema cannot specify *how pressing* and *rotating* motions must happen. Instead the *screw* plan *knows* it must simultaneously run the *press* and *rotate* plans.

A. Reasoning On Fully Parametrized Plan Schemata

From a fully parametrized plan schema our Prolog-based reasoning mechanism extracts designators for: actions, objects, and locations. In particular for the fully parametrized pipetting plan schema the reasoning mechanism extracts the action designator for pipetting action, the object designators for pipette and containers and the location designators relative to them. Code Excerpts 3 - 4. Designators are symbolic descriptions. Syntactically they have the form of a set of attribute-value pairs. Semantically they start existing

$$((\langle attr_0 \rangle \ \langle val_0 \rangle) \ \ldots (\langle attr_n \rangle \ \langle val_n \rangle))$$

as underspecified descriptions for each entity involved by NL instruction and needs a representation. The semantics of a designator gives designator's type. While the control system is running those symbolic representation grow complex incorporating more details about the entities they are referring to.

Code Excerpt 3 Object Designators

```
(test-tube
  (type Container)
  (size 500ml)
  (contains NaOH)
  (has-a
    (cover
      (type cap)
      (color blue))))

(pipette
  (type instrument)
  (capacity 10ml)
  (has-a button-designator)
  (has-a effector-designator)
```

Code Excerpt 4 Location Designators

```
(above
  test-tube-designator)

(inside
  bottle-NaOH-designator)
```

Pipetting action designator from Code Excerpt 2 states *what* pipetting action needs in terms of classes of entities - specifically it needs two entities of type *Container* and an entity of type *Instrument*. The difference between the pipetting plan schema and the pipetting action designator resides on their different domains of definition. First of them is defined over the set of action roles and the second is defined over the set of symbolic features.

Test tube designator Code Excerpt 3 states that it is of *type Container*, having a *size* of *500ml*, contains *NaOH*, and *has-a cover* of *type cap* and *color blue*. Robot's *Object Recognition System* detailed in Section IV accepts the vague and symbolic object designators of test tube and returns it enriched with more perceived details like for example test tube's 6D pose.

Location designators are defined relative to object designators. They behave like space quantifiers and refer to different regions around objects. *Above* and *inside* are two location designators. They are defined relatively to at least one object. In Code Excerpt 4 they refer the spatial region above the test tube respectively the spatial region inside the bottle containing the chemical compound NaOH.

B. Plan Execution

The reasoning mechanism infers from pipetting action designator that the *pipet* plan, depicted in Code Excerpt 5, is the most competent to perform it. The pipet plan takes as arguments four designators which symbolically describe the *source* holding the liquid from which the *amount* must be transfered into the *destination* by using the *instrument*. Inside its body, the *pipet* plan coordinates sequentially another two plans which *aspirate* a specific *amount* of liquid into the *instrument* and *dispense* it into the *destination*. At its turn the *aspirate* plan coordinates other simpler plans which *move* an object, *press* an object part respectively *release* an object part. In order to move *instrument*'s *effector* (pipette's tip)

Preprint submitted to 2015 IEEE/RSJ International Conference
on Intelligent Robots and Systems. Received March 4, 2015.

Code Excerpt 5 Pipetting Plan

```
(def-plan pipet (source instrument amount destination)
  (seq
    (aspirate (source instrument amount))
    (dispense (instrument amount destination))))

(def-plan aspirate (source instrument amount)
  (seq
    (recognize source)
    (move (object-part effector instrument)
          (above source))
    (recognize (object-part button instrument))
    (press (object-part button instrument))
    (move (object-part effector instrument)
          (inside source))
    (release (object-part button instrument))
    (move (object-part effector instrument)
          (above source))))
```

the *object-part* quantifier is used to cast the effector as an object and give it as actual parameter to the *move* plan call. Internally the *move* plan figures out the relation between object's frame to be moved and object's grasping points. Taking into account this relation the plan is appropriately parametrizing the controller to perform the right motions.

1) Plan Language: For coding the *pipet* plan we used CRAM Plan Language (CPL) [8] which reimplements and extends RPL [9]. CPL's control structures are designed to allow reasoning about the plan and revising it in case a failure is detected. Plans implemented in CPL can be more than a sequence of atomic actions. They can run concurrently, in loops, they can be synchronized and they benefit of failure handling mechanism. Reasoning on plans can be done without a complete understanding of a whole plan because CPL's control structures support annotation.

2) Plan Library: Top-down the plan library contains task abstract but action specific plans. Bottom-up it contains hardware specific plans which communicate with robot's object recognition system and controllers via ROS [10] middleware. *pipet*, *aspirate* or *screw* are just few action specific plans. *recognize*, *move* or *rotate* are other few hardware specific plans. Action specific plans build on top of hardware specific plans.

3) Reactive Execution Engine: At execution time the *pipet* plan is run as a normal control program. In the first phase the reactive execution engine queries the object recognition system, detailed in the next section by sending vague object designators and receiving them enriched with more details about recognized objects. In the second phase, before triggering robot's controllers, the reactive execution engine asks the geometric reasoning module [11] to check if the intended manipulations are feasible. The geometric reasoning module temporal projects the requested manipulations and analyzes them. If an issue is detected then the plan gets the chance to fix it. If the geometric reasoning doesn't return any issue then in the fourth phase the cartesian controller is employed to move robot's arms and perform the motions requested. For future experiments we plan to employ either a motion planner either more flexible controllers [12].

4) Spatial Reasoning: At plan's runtime within robot's specific context, all symbolic location designators must be converted into numerical values understandable by robot's controllers. The geometric reasoning mechanism associates a three dimensional probability distribution to each location designator and draws a sample out of it. For moving pipette's tip inside bottle which contains sodium hydroxide, the geometric reasoning mechanism draws a sample from the probability distribution describing the volume inside the bottle. Based on this sample the *move* plan will parametrize robot's arm controllers such that they move pipette's tip in the sampled three dimensional value. Besides converting symbolic descriptions to numerical values the geometric reasoning mechanism has other more powerful functionalities like asserting if the current manipulation of an object will obstruct future manipulations involving other objects or asserting if the current manipulation will leave the environment into a stable state.

5) Cartesian Controller: We focus our experiments on observing how robots can perform NL instructions, more precisely on bridging NL understanding with robot's control programs. In order to move robot's arms, for a moment, we chose the simplest approach of using a inverse kinematics on top of a joint controller. For future experiments we are integrating a more flexible controller which uses defined constraints over a set of features.

6) Semantic Logging: When running a given plan the reactive execution engine signals a multitude of execution context characteristics like: plan's goals, the relations between the plan being run and the other sub-plans called by it or pieces of sensor data which influenced robot's decisions [3]. All descriptions are synchronized based on a time stamp.

7) OpenEASE: collects all descriptions generated by the semantic logging module and makes them available to other robots [4]. OpenEASE is equiped with inference tools which allow reasoning on this data and answering queries regarding to what did the robot see, why, how, did the robot behaved.

VI. Evaluation

Our robot took part in *Ocean Sampling Day* [13] an event organized with the aim of indexing all DNAs from planetary ocean. Ideally it should have performed the entire procedure of DNA extraction on the samples collected within this event, but because of procedure's size we had to limit our experiments to just few steps. For testing the pipeline proposed in Section II, from *DNA Extraction Procedure* we selected the neutralization instruction and from *Pipette Usage* we selected the two instructions for pipetting an amount of liquid, all of them stated in NL. *PRAC* system successfully attached an action role to each instruction word and inferred that the pipetting plan schema is the most appropriate to be parametrized with the specific details coming from instructions' words. The Prolog-based reasoning mechanism successfully extracted the symbolic descriptions of pipetting action, for the containers involved and the necessarily instrument and inferred that the *pipet* plan is the most competent to perform the pipetting action. When running the reasoning mechanism on the extracted pipetting action designator only the pipetting plan is identified as the most

appropriate for performing the pipetting action. In future experiments we want to test whether the reasoning mechanism is able to infer an ensemble of plans which combined will should perform given action designator, be it pipetting. When executing the pipetting plan, object recognition system successfully recognized all objects involved based only on their symbolic description Figure 6. For representing the type of knowledge the robot needs in order to press pipette's button such that right amount of liquid is released we decided to use the *KnowRob* [14] knowledge processing system for our future experiments. The cartesian controller behaved well for simple manipulations but we expect it to be overtaken in our future experiments.

VII. Related Work

The system proposed in [15] probabilistically maps NL instructions into a set of robot primary actions and obtains the sequence of manipulations from planning in this set. The system [16] turns NL commands into a more structured representation and learns a probabilistic graphical model to associate the structured representation to a plan inferred from the set of groundings: objects, locations, actions. For training the probabilistic model people are shown a task happening inside a simulator and are asked to state in NL commands which correspond to task's requirements. The system described in [17] obtained very promising results by building, at learning time. a conditional random field over the set of NL commands and using it at runtime for minimizing an energy function over new commands. So far these systems skipped the problem of *understanding* NL instructions and focused more on correctly *associating* NL instructions to robotic primitive actions.

Our approach is very similar with [18] where the two robots read instructions from web, turn them into executable plans such that at runtime they collaborate in making a pancake.

If we shortly look at chemical experiments from the perspective of autonomously designing and testing hypotheses respectively interpreting the obtained results, Adam, the robot scientist [19] obtained remarkable results. Now looking at how the results of chemical experiments are recorded on Electronic Laboratory Notebooks [20] and how easy are they shared through *Chemical Semantic Web* we believe autonomous mobile robots are capable of using these results and produce new ones.

VIII. Conclusions

In this paper we present the control system of an intelligent autonomous robot which is able to understand NL instructions and infer and run the most competent control program for performing them. For each instruction the NL Processing System successfully inferred the implicit knowledge and assembled reach instruction representation. From this representation the reasoning mechanism extracted symbolic descriptions for action, objects, locations and the most competent control programs to coordinate robot's required motions. The experiments we made suggested us that the reasoning mechanism is capable of inferring more than one best control program but instead it is capable of inferring how to combine more control programs into more competent one. The results of these experiments will be published in a future paper. When the inferred control program was run, the reactive execution engine competently coordinated the object recognition system, the geometric reasoning system and robot's controllers to accurately recognize the required objects and competently manipulating them. The control programs contained in the plan library proved to be very flexible and highly parametrizable. Overall the entire proposed control architecture proved itself to be very scalable. The used symbolic mechanism is compatible with newly developed semantic web tools for chemistry. The results of our future experiments will report how can the chemistry semantic web be made available to intelligent robots. The semantic logging mechanism recorded all robot experiences and OpenEASE, the web-based knowledge base for robots makes them available to other robots.

References

[1] M.-C. De Marneffe, B. MacCartney, C. D. Manning, *et al.*, "Generating typed dependency parses from phrase structure parses," in *Proceedings of LREC*, vol. 6, 2006, pp. 449–454.

[2] "WordNet," 2008, `wordnet.princeton.edu`.

[3] J. Winkler, M. Tenorth, A. K. Bozcuoglu, and M. Beetz, "CRAMm – memories for robots performing everyday manipulation activities," *Advances in Cognitive Systems*, vol. 3, pp. 47–66, 2014.

[4] M. Beetz, M. Tenorth, and J. Winkler, "Open-EASE – a knowledge processing service for robots and robotics/ai researchers," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015, accepted for publication.

[5] D. Nyga and M. Beetz, "Everything robots always wanted to know about housework (but were afraid to ask)," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

[6] M. Richardson and P. Domingos, "Markov Logic Networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[7] M. Beetz, F. Balint-Benczedi, N. Blodow, D. Nyga, T. Wiedemeyer, and Z.-C. Marton, "RoboSherlock: Unstructured Information Processing for Robot Perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015, accepted for publication.

[8] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 18-22 2010, pp. 1012–1017.

[9] D. McDermott, "A Reactive Plan Language," Yale University," Research Report YALEU/DCS/RR-864, 1991.

[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *In IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12–17 2009.

[11] L. Mösenlechner and M. Beetz, "Fast temporal projection using accurate physics-based geometric reasoning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6–10 2013, pp. 1821–1827.

[12] G. Bartels, I. Kresse, and M. Beetz, "Constraint-based movement representation grounded in geometric features," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Atlanta, Georgia, USA, October 15–17 2013.

[13] Micro B3. (2014) Ocean sampling day. [Online]. Available: www.microb3.eu/osd

[14] M. Tenorth and M. Beetz, "KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 5, pp. 566 – 590, April 2013.

Preprint submitted to 2015 IEEE/RSJ International Conference
on Intelligent Robots and Systems. Received March 4, 2015.

[15] Mario Bollini, Jennifer Barry, and Daniela Rus, "BakeBot: Baking Cookies with the PR2," in *The PR2 Workshop, from International Conference on Intelligent Robots and Systems (IROS)*, 2011.
[16] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation." in *AAAI*, 2011.
[17] Dipendra K Misra, Jaeyong Sung, Kevin Lee, Ashutosh Saxena, "Tell Me Dave: Context-Sensitive Grounding of Natural Language to Mobile Manipulation Instructions," in *Robotics: Science and Systems (RSS)*. IEEE, 2014.
[18] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic Roommates Making Pancakes," in *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.
[19] R. D. King, "The automation of science," *Science*, vol. 324, no. 5923, pp. 85–89, April 3, 2009.
[20] C. L. Bird, C. Willoughby, and J. G. Frey, "Laboratory notebooks in the digital era: the role of elns in record keeping for chemistry and other sciences," *Chem. Soc. Rev.*, vol. 42, pp. 8157–8175, 2013. [Online]. Available: http://dx.doi.org/10.1039/C3CS60122F

Preprint submitted to 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 4, 2015.