Cloud-based Probabilistic Knowledge Services for Instruction Interpretation

Daniel Nyga and Michael Beetz

Abstract As the tasks of autonomous manipulation robots get more complex, the tasking of the robots using natural-language instructions becomes more important. Executing such instructions in the way they are meant often requires robots to infer missing, and disambiguate given information using lots of common and commonsense knowledge. In this work, we report on *Probabilistic Action Cores* (PRAC) [21] – a framework for learning of and reasoning about action-specific probabilistic knowledge bases that can be learned from hand-labeled instructions to address this problem. In PRAC, knowledge about actions and objects is compactly represented by first-order probabilistic models, which are used to learn a joint probability distribution over the ways in which instructions for a given action verb are formulated. These joint probability distributions are then used to compute the plan instantiation that has the highest probability of producing the intended action given the natural language instruction. Formulating plan interpretation as a conditional probability is a promising approach because we can at the same time infer the plan that is most appropriate for performing the instruction, the refinement of the parameters of the plan on the basis of the information given in the instruction, and automatically fill in missing parameters by inferring their most probable value from the distribution. PRAC has been implemented as a web-based online service on the cloud-robotics platform openEASE [7].

1 Introduction

In artificial intelligence, the problem of interpreting instructions is mostly approached by applying automated action planning methods in order to generate plans for tasks. However, the plans generated by such systems are too abstract for compe-

Daniel Nyga and Michael Beetz
Institute for Artificial Intelligence, University of Bremen, Germany, e-mail: {nyga,beetz}@cs.uni-bremen.de

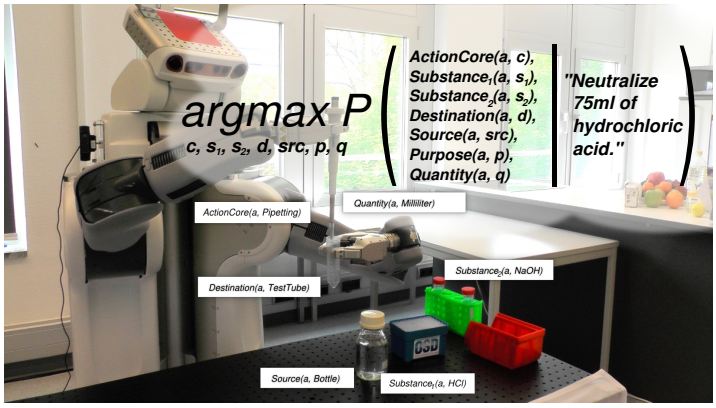

Fig. 1 Exemplary reasoning task in PRAC for interpreting a NL instruction.

tent execution by robots since they merely contain what is *given* but not what is *necessary*. A promising alternative is to generate plans for tasks from natural-language instructions that humans write for humans. Such instructions are available in abundance in the world-wide web at websites like wikihow.com, ehow.com, and many others. Instructions written for humans are more informative than automatically generated action plans because they often describe *how* actions have to be performed to bring about the desired effects, they talk about what can go wrong, and give additional hints. For robotic agents it makes sense to consider instruction understanding to be the computational problem of inferring how the agent could (successfully) perform the instruction. This problem formulation is substantially different to the problem of text understanding for question answering or machine translation. In those reasoning tasks, the vagueness and ambiguity of natural-language expressions can often be kept and translated into other languages. In contrast, robotic agents have to infer missing information pieces and disambiguate the meaning of the instruction in order to perform the instruction successfully.

Thus, if a robotic agent is tasked with the instruction *"neutralize 75ml of hydrochloric acid"*, for instance, the robot has to infer that neutralization requires to add a some amount of base substance to the hydrochloric acid. It also has to infer that this means that some amount of the base substance has to be transferred from the container which it is contained in into the container that holds the acid substance. Finally, because the amount is small and accurately specified the adding step should be performed through a pipetting action. As another example, consider the two instructions "fill the kettle with water" and "fill a cup with coffee." Though the syntactic structure of the sentences as well as their semantics are identical, they fundamentally differ with respect to execution. Filling a kettle with water can be achieved by using the tap, whereas filling a mug with coffee implies a pouring motion from a coffee pot into a cup. In other words, understanding a natural-language instruction for robot execution requires *appropriate interpretation and completion*.

For the purpose of this paper we consider robotic agents that are equipped with a plan library that contains parameterizable plans for action verbs, which have to be refined according to a given instruction. In this case instruction understanding can be realized by retrieving the plan corresponding to the action required by the instruction and by constraining its parameterization according to the instruction.

To deal with the incomplete and ambiguous nature of natural-language instructions, we phrase the problem as a probabilistic reasoning problem, namely that of finding the most probable 'executable' refinement of the respective general plan given the natural-language instruction as evidence:

$$\arg\max_{\text{plan}} P\left(\text{intended(plan)} \,\middle|\, \begin{array}{l}\text{“neutralize 75ml}\\ \text{of hydrochloric acid”}\end{array}\right).$$

To perform this inference task we equip the robot with a joint probability distribution over the source, destination, the object acted on, the tool to be used, and other action roles for each action verb. To this end, we introduce the notion of *action cores*, which are conceptualizations of action verbs that represent formal specifications of actions and their parameters that are capable of interfacing the plans on a symbolic, linguistic level. The resulting probabilistic first-order knowledge base of action cores and their respective action roles is called *probabilistic action cores* (PRAC). PRACs can be used to perform disambiguation and completion of vague, underspecified natural-language (NL) sentences and thus are suitable for NL instruction interpretation.

An example of such an inference process is depicted in Figure 1 which will also be the running example for our paper. We have equipped the robot with a plan library including plans for pouring and pipetting, among many others. The parameters of the plans for pouring and pipetting are the *theme* of the action, meaning the stuff to be transported from one place to another one, the *source* of the stuff, and the *destination* of the stuff. The problem of instruction interpretation for robot execution can now be formulated as the reasoning task of inferring the most probable plan (*action_core(a,c)*) and the most probable refinement of the formal plan parameters (source, destination, theme) given the natural language instruction *"neutralize 75ml of hydrochloric acid"*.

This is a very elegant and general formulation of instruction interpretation because by doing the inference task on a joint probability distribution over action instructions we can at the same time infer the plan that is most appropriate for performing the instruction, the refinement of the parameters of the plan schema on the basis of the information given in the instruction, and automatically fill in missing parameters by inferring their most probable value from the distribution.
The key contributions of this paper are the following:

- We formalize PRAC and the computational problem of inferring the most probable executable action instruction.
- We show how PRACs can be realized as Markov logic knowledge bases and learned from few examples.

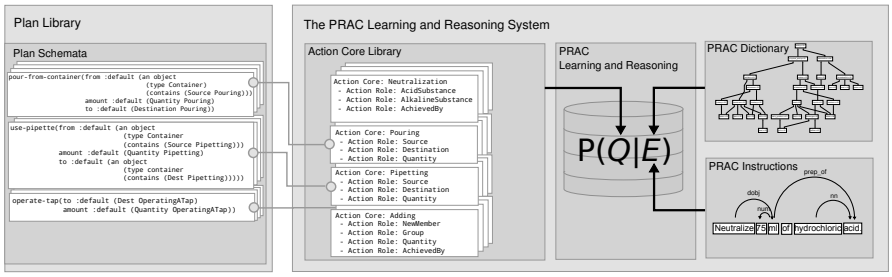

Fig. 2 Key concepts of the framework and their role in inferring the most probable executable instruction.

- We show how the problem of inferring the most probable executable action instruction can be implemented to yield effective solutions as full size first-order probabilistic reasoning problems.

In the remainder of this paper we proceed as follows. We start with an introduction to the PRAC framework and detail its conceptual components. Subsequently, we give a formal definition of the reasoning tasks that PRAC addresses and describe our approach for tackling them. Then we discuss the state-of-the-art in instruction interpretation for robot commanding and conclude our work.

2 Conceptual Framework

The PRAC framework for translating natural-language instructions into abstractly parameterized robot action plans is depicted in Figure 2. Its main components are the **PRAC plan library**, the **PRAC knowledge base**, and the **PRAC dictionary**. In a nutshell, the roles of these components are the following ones.

The **PRAC dictionary** provides all possible meanings of all the words that can occur in a NL instruction to be executed by a robot. The meanings are concepts in an ontological knowledge base defined in the dictionary *WordNet* [12], which comprises more than 117,000 concepts. For example, the possible meanings of 'cup' in the PRAC dictionary include a specialization of a physical object and, more specifically, a container object, an amount specification, and a trophy (see also Figure 3).

The **PRAC knowledge base** contains a collection of action verb-specific knowledge bases, called **action cores**, that represent how possible action instructions for a given action verb can be constructed on a conceptual level. For example, we can formalize a pouring action on a concept level in terms of conjunctions of logical assertions over the predicates *action_core(a, Pouring)*, *theme(a, t)*, *source(a, s)*, *destination(a, d)*, etc. The assertion *theme(a, t)* states that the theme of action a is of the type t, i.e. the entity which is poured. The parameters t, s and d are concepts in the PRAC dictionary.

Action-specific knowledge bases are then trained with a set of instructions stated in first-order logic in order to learn a joint probability distribution over predicate instantiations, which is induced by the given set of instructions. These distributions are called the **probabilistic action core (PRAC)**. The learned distribution represents correlations between the concept restrictions of the parameters in instructions with respect to an action verb. For example, the PRAC of Pouring could entail that if the Theme of a pouring action is the concept wine then it is more likely that the source for the pouring action will be an instance of the concept bottle and the destination an instance of the concept glass. Conversely, if the Theme is of the concept water, then the source is more likely to be a tap.

Finally, the **PRAC plan library** contains action specific plans. PRAC plans are equipped with plan signatures following the 'design-by-contract' principle: The plan signature specifies the formal parameters of the plan, the concept restrictions for each parameter and how the respective plan parameter can be computed from the PRAC knowledge base. For example, the signature of the plan for a pouring action looks as follows:

```
pour-from-container(from :default (an object
                                    (type container.n.01)
                                    (contains (Pouring Theme))
                    amount :default (Pouring Quantity)
                    to :default (an object
                                    (type container.n.01)
                                    (contains (Pouring Goal))))
```

The plan schema specifies that the from parameter has to be a specialization of the concept container.n.01 and that the from parameter can be retrieved from the PRAC knowledge base of Pouring by retrieving the value of the role Theme of Pouring. Likewise, the amount parameter can be obtained by querying for the Quantity predicate.

It is required that all formal parameters of the plan are linked to roles in the respective action core. By providing a plan signature, the designer of the plan guarantees that for all plan refinements that satisfy the concept restrictions of the individual parameters, executing the plan generates meaningful behavior. 'Meaningful' here means that the plan generates behavior that makes sense but is not required to succeed. For a pouring action, for instance, the plan tells the robot to grasp the source container, to hold it above the destination and to tilt it. However, the execution of the parameterized plan hazards failures caused by inappropriate motor control or inaccurate perception, such as spilling the liquid because the container is held too high, off center, or the pouring angle is too steep. This requires that all parameters needed to call sub-plans are computed by the plan and no call to a sub-plan contains undefined parameters, which would cause the control system to crash.

The plans themselves are considered as black boxes in PRAC reasoning. Plan execution systems that can handle such qualitative, symbolic constraints on parameters include RAP [13] and PRS [14]. If deeper reasoning about the ramifications of actions is necessary, the CRAM [19, 6] executive provides reasoning methods that translate qualitative constraints into PROLOG queries that use sampling and back-

predicate	meaning
$det(w_1, w_2)$	w_2 is the determiner of w_1
$dobj(w_1, w_2)$	w_1 is the direct object of w_2
$advmod(w_1, w_2)$	w_1 is an adverbal modifier of w_2
$prep\_with(w_1, w_2)$	w_1 and w_2 are connected by 'with'
$nn(w_1, w_2)$	w_1 and w_2 are a compound noun
$has\_pos(w, p)$	w has the part-of-speech p

concept	definition
cup.n.01	a small open container usually used for drinking; usually has a handle
cup.n.02	cupful, the quantity a cup will hold
cup.n.08	a large metal vessel with two handles that is awarded as a trophy to the winner of a competition

Fig. 3 *Left:* Selection of logical predicates representing syntactic structure of words in a sentence. A comprehesive list can be found in [9] *Right:* Selection of different meanings of the word 'cup' obtained from WordNet.

tracking to find parameter instantiations satisfying these constraints. Kinds of such parameters include e.g. action effects, visibility, reachability and the like.

Using the components of the PRAC system introduced above, the computational process for computing the most probable executable instruction operates as follows: In a first step, a given natural-language instruction ι is translated by a natural-language parser into a logical representation of the instruction's syntactic structure I, which we call a PRAC *instruction*. The PRAC instruction I is then interpreted by inferring the meaning and semantic role of the individual syntactic structures and missing information pieces using the PRAC *dictionary* and the action core itself. This interpretation process results in the *most probable executable instruction* of ι. In the remainder of this section we will describe in more detail the concepts and components that learning and reasoning about action cores is built upon.

2.1 PRAC *Instructions*

A PRAC instruction I is a set of assertions about the grammatical relations referring to the constituents of a natural-language instruction and their syntactic structure. These grammatical relations are represented by predicates including the small selection listed in Figure 3. They are obtained for any sentence in natural language by a parser like the Stanford parser [8]. Using these predicates, a natural-language instruction such as $\iota =$ "neutralize the hydrochloric acid with sodium hydroxide", for example, is transformed into the logical assertions I,

$$\begin{aligned}
&\textit{dobj}(\textit{neutralize-1}, \textit{acid-4}) \quad \textit{has_pos}(\textit{neutralize-1}, \textit{VB})\\
&\textit{det}(\textit{acid-4}, \textit{the-2}) \qquad\quad \textit{has_pos}(\textit{acid-4}, \textit{NN})\\
&\textit{nn}(\textit{acid-4}, \textit{hydrochlroic-3}) \; \textit{has_pos}(\textit{hydroxide-7}, \textit{NN})\\
&\textit{nn}(\textit{hydroxide-7}, \textit{sodium-5}) \; \textit{has_pos}(\textit{sodium-6}, \textit{NN})\\
&\textit{prep_with}(\textit{neutralize-1}, \textit{hydroxide-7}), \qquad\qquad (1)
\end{aligned}$$

which we denote by $\mathscr{I}(\iota)$. These syntactic dependencies indicate that the second word 'the' depends on the fourth word 'acid' as a determiner, 'hydrochloric' and 'acid' represent a compound noun, which forms the direct object of the word 'neu-

tralize', which is connected to the word hydroxide via the preposition 'with'. The syntactic structure of the instruction thus forms a relational database that serves as evidence in a probabilistic relational model. For a more detailed and exhaustive description of the syntactic dependencies, we refer to [9].

2.2 PRAC *Dictionary*

The PRAC dictionary is a set of logical assertions that assign meaning (word senses) to words. It is filled with word senses ('synsets') from the online dictionary WordNet[1] (see Figure 5). The word senses are organized in a taxonomy given by a directed acyclic graph, which we denote by the relation $\sqsubseteq$, i.e. $c_1 \sqsubseteq c_2$ denotes that the concept c_1 is a specialization of concept c_2.

In the PRAC dictionary, a word w is assigned a particular meaning m by means of a set of logical assertions $has\_sense(w,m)$ and $is\_a(m,c)$ $\forall c$ $m \sqsubseteq c$, where $has\_sense(w,m)$ states that the word w has the sense m in the WordNet dictionary and the is_a predicate is the transitive closure of m in $\sqsubseteq$. The PRAC dictionary also provides a function $\mu\colon W \times P \mapsto \mathscr{P}(\top)$, which returns the set of all possible meanings of a word given its part of speech, where W denotes the set of all words, P is the set of all parts of speech, $\top$ is the set of all concepts in $\sqsubseteq$ and $\mathscr{P}(\cdot)$ denotes the power set.

Words can have multiple meanings causing ambiguity in NL instructions. Consider, for example, the terms 'cup' and 'milk' and their meaning in the two instructions "fill a cup with milk" and "add a cup of milk." In the former case, 'cup' refers to a drinking mug, a physical object that can hold milk. In the latter case, it rather refers to a measurement unit specifying the amount of milk to be added. Though this semantic difference may seem subtle, correctly distinguishing between word meanings is crucial for successfully performing the actions. Thus, in finding an appropriate interpretation of an instruction, selecting the most appriopriate word meanings is a necessity.

2.3 PRAC *Knowledge Base*

The PRAC knowledge base is the central component of the PRAC system. It contains a library of data structures, which we call **action cores**. An **action core** is the conceptualization of an action which constitutes an abstract event type and assigns an action role to each entity that is needed in order to successfully perform the respective action.

More formally, an action core AC is defined as a tuple $\langle A,R\rangle$, where A is the globally unique name of the action core and $R = \{r_{A_i}\}_{i=1}^{n_A}$ is an indexed set of its associ-

[1] All concept names refer to concept names provided by the NLTK toolbox (http://www.nltk.org)

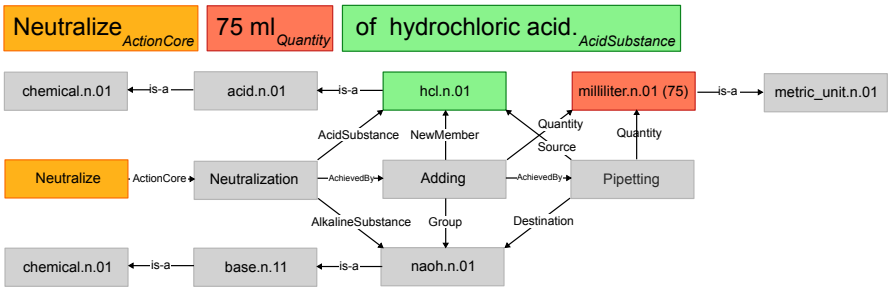

Fig. 4 Exemplary action instantiation for the instruction "Neutralize 75 ml of hydrochloric acid." Taxonomy paths (*is-a*) are truncated for better readability.

ated action roles. For an interpretation x of a PRAC instruction I, $interpretation(x,I)$, the following holds:

$$action\_core(x,A) \rightarrow \exists c_1,\ldots,c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(x,c_i),\ c_i \in \top \tag{2}$$

The right side of the implication in (2) ensures that every instantiation of an action core must have a complete assignment of its action roles to concepts in $\top$, otherwise it is not an instance of the action core. However, being able to assign all roles of an action core does not imply that it must have an instance in x. Equation 2 defines the space $\mathscr{X}$ of possible interpretations of an instruction. A graphical representation of one particular interpretation of the instruction "neutralize 75 ml of hydrochloric acid" is shown in Figure 4: There are instances of the three action cores Neutralization, Adding and Pipetting with their respective roles assigned a concept. The set of action cores and our definition of an interpretation can thus be regarded as a template for constructing a graphical model of interpretations like the one in Figure 4.

An example of the action core Pouring and its action roles was already given above. In that context, the action core has a direct mapping to a plan schema in the PRAC plan library and its action roles Source, Destination and Theme interface the formal parameters of the plan schema. As another example, consider the action core Neutralization, representing the process of causing a chemical substance to take a neutral pH-value by combining it with some other substance. For the chemical reaction itself, there must always be two components reacting, an acid and a base. The corresponding action core Neutralization thus is attached two action roles, AcidSubstance and AlkalineSubstance. It is important to note that within PRAC, the domains of action roles and their corresponding parameter slots in the plan schemata are given by the set $\top$ of all concepts from the ontological knowledge base in the PRAC dictionary. This ensures that all symbols have the same semantics across the different components of PRAC, the syntactic representation in the PRAC instructions, the semantic action representation of action cores as well as the plan schemata.

There is an action core for every verb in the PRAC dictionary that represents a meaningful action. However, there are action cores that do not have a direct correspondence to a plan schema because they do not represent actions that are directly

executable but are subject to further reasoning. Neutralization is an example of such an action core: It is not an executable action as such, but rather describes a chemical process that is triggered by Adding one substance to the other. Adding itself is an action core representing the process of making a new member part of an existing group. It has three action roles, namely the Group, the NewMember and the Quantity. The Adding action core, however, also represents a process that can be achieved in very different ways depending on the context and the objects involved. For example, "add one liter of water" could be achieved by using the tap or pouring from one container to the other, whereas "add one milliliter of water" should be performed by using a pipette. Conversely, "add a pinch of salt" can be done by using a salt cellar. Such an action core A that does not have a direct mapping to an executable plan schema has attached a designated action role AchievedBy(A, A'), which is assigned another action core A' that represents the most likely refinement of the action represented by A.

The goal in natural-language instruction understanding is now to find the *most probable interpretation under the instruction given as evidence*. Therefore, the PRAC knowledge base has a conditional probability distribution over all action cores and their action roles, conditioned on the PRAC dictionary and the PRAC instructions, as depicted in Figure 2,

$$P\left(\begin{array}{l} action\_core(x,A) \rightarrow \\ \quad \exists c_1,\ldots,c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(x,c_i) \end{array} \middle| \top, \mathscr{I} \right). \tag{3}$$

We call (3) the **probabilistic action core** (PRAC). The probabilistic action core is a first-order probabilistic knowledge base about actions and their parameterizations that is used to disambiguate, interpret, complete and refine NL instructions.

2.4 Examples

The probabilistic action core can be used to resolve ambiguity and to complete an instruction to the most plausible action specification, based on what is given by the instruction. In the following, we will illustrate the usage of the PRAC distribution by means of three simple exemplary queries[2].

Action role assignment. PRAC can be queried for the most likely assignment of roles for a given set of objects with respect to a particular action core. Consider an instruction, such as "add 3 drops of sodium hydroxide." There are two objects o_1 and o_2 in the instruction given by the concepts *naoh.n.01* and *drop.n.02* in the PRAC dictionary. In context of the Adding action core, one can solve for the most probable assignment of the action roles attached to Adding, i.e.

[2] We are using a slightly modified notation, which technically does not precisely fit the previous formulations. We think this simplified notation better supports the understanding of reasoning considered in this paper.

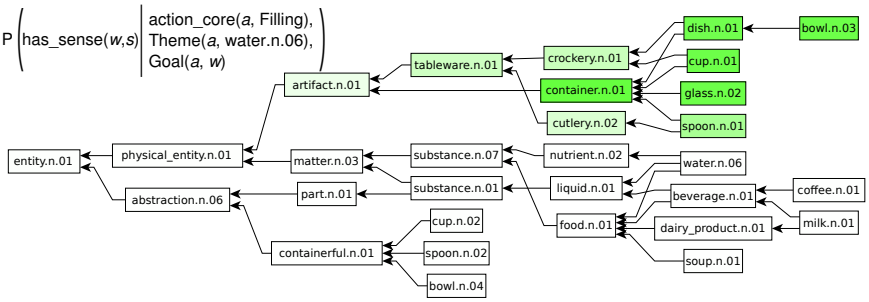

Fig. 5 Conditional distribution over an excerpt of the PRAC taxonomy structure for containers, substances and measuring units for an entity w taking the Goal role of the Filling action core.

$$\underset{o_1',o_2',o_3' \in \{o_1,o_2,\perp\}}{\arg\max} \; P\left(\begin{array}{l} \mathit{Quantity}(o_1'), \\ \mathit{NewMember}(o_2'), \\ \mathit{Group}(o_3') \end{array} \middle| \begin{array}{l} \mathit{is\_a}(o_1, \mathit{drop.n.02}), \\ \mathit{is\_a}(o_2, \mathit{naoh.n.01}) \end{array} \right) = \left\{ \begin{array}{l} o_1' = o_1, \\ o_2' = o_2, \\ o_3' = \perp \end{array} \right\},$$

where $\perp$ denotes the *null* assignment. In this example, the Quantity role has been assigned the object *drop.n.01*, the NewMember role the object *naoh.n.01* and the Group could not have been assigned any of the objects mentioned in the instruction. Note that this arg max solution is not a proper interpretation of the instruction in the notion from above, because there is no Group specified.

Action role completion. In order to fill missing role assignments such as the Group in the previous example, one can solve for a different arg max query. Consider the instruction "neutralize the hydrochloric acid" and suppose we have already assigned the object *hcl.n.01* the role AcidSubstance of the Neutralization action core. According to its definition, there must be the role AlkalineSubstance assigned to some concept, which is not given in the instruction. In order to infer its role assignment, we introduce a Skolem constant s that hypothetically fills the missing role slot of AlkalineSubstance. Since the taxonomy relation of the PRAC dictionary is included in the PRAC distribution, one can query for the most probable type of s:

$$\underset{c \in \top}{\arg\max}\, P\left(\mathit{is\_a}(s,c) \middle| \begin{array}{l} \mathit{is\_a}(\mathit{hcl}, \mathit{hcl.n.01}), \\ \mathit{AcidSubstance}(\mathit{hcl}), \\ \mathit{AlkalineSubstance}(s) \end{array} \right) = \mathit{naoh.n.01},$$

which means that sodium hydroxide (NaOH) is the most probable alkaline counterpart for the Neutralization of hydrochloric acid (HCl).

Joint distributions over taxonomies. One of the key features of PRAC is the ability to perform reasoning about unmodeled concepts, i.e. concepts that have not been seen during learning. This enables (1) a compact representation of knowledge, (2) efficient transfer of the learnt knowledge to new situations and (3) filling missing information pieces in underdetermined action specifications. Figure 5 shows an example of a conditional distribution over concepts in the PRAC taxonomy for potential Goals of a filling action: From all concepts, specializations of containers gain

highest probability, which reasonably reflects our intuitions about a typical filling action. Since PRAC maintains joint distributions over the action roles and the concepts in the PRAC dictionary, we can compute any conditional distribution given any evidence, which enables context-sensitive completion of actions like in the previous example. Such out-of-domain inference tasks are implemented using the FUZZY-MLN reasoning framework [22].

3 The PRAC Learning and Reasoning System

In this section, we describe in more detail how reasoning is implemented in the PRAC framework. We first depict the basic ideas of learning and inference in PRAC, address the issues of learning and present the processing pipeline for performing inference about interpretations and completions of natural-language instructions. There are two key paradigms in the PRAC reasoning system.

Learning by generalization. Humans are capable of learning rapidly and flexibly how to use different words in different situations by only having seen very few examples. They have available an efficient apparatus for generalization, which allows them to abstract away from a very small set of specific instantiations to more generic patterns of everyday situations that we often encounter in their 'typical' form. Consider the example of a 'filling' action. From hearing just a few specific instances of that action verb, e.g., "fill a pot with water" and "fill a cup with milk", humans are capable of generalizing to a stereotyped pattern like "fill a container with a liquid." This kind of generalization is both powerful and efficent since, on the one hand, it enables compact representation of knowledge and on the other hand, it allows to treat new, unseen examples in a meaningful way.

Inference by specialization. Reasoning about new, unseen situations is done by selecting one or more generic patterns that best fit the new situation and by adapting them to reality as necessary in order to come up with an instantiated representation which is as specific and unambiguous as possible. In the example from above, an instruction like "fill a glass with juice," for instance, is matched against the generic 'filling' action and is adapted accordingly by inspecting the conceptual subsumption of the terms 'juice', which corresponds to the liquid being poured and 'glass', which constitutes the goal container of the filling action.

PRAC implements these two paradigms in a coherent probabilistic framework, which automatically finds abstractions of common situations as illustrated in the above examples by exploiting the semantic similarities of concepts in the taxonomy graph. These abstractions reasonably reflect human intuitions of how specific terms are to be used in certain situations. These principles of abstraction and generalization from examples also constitute cornerstones of human cognition [26, 3, 17]. As an implementational framework, we use Markov logic networks (MLN) [23] to encode the knowledge about action cores, their action roles, the PRAC dictionary and the PRAC instructions, which is a powerful knowledge representation formalism that combines first-order logic with probability theory.

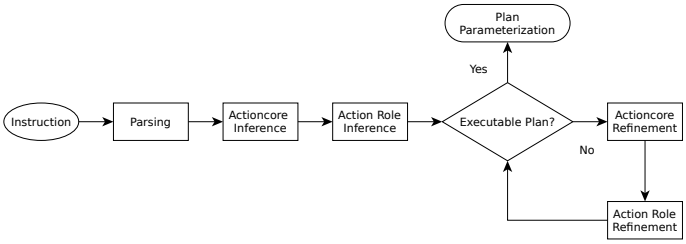

Fig. 6 Flow diagram of the PRAC reasoning pipeline with the steps (1) NL parsing (2) action core identification (3) action role identification (4) checking for an executable plan schema attached to an action core and (5) action core refinement.

3.1 Reasoning

The probabilistic first-order knowledge base in (3) for solving inference problems of the form $\arg\max_Q P(Q \mid E)$ has an enormous size. It contains at least the cross product of all possible word meanings squared and roles where the set of the possible word meanings include all possible meanings of the words that occur in the training data plus the number of their superconcepts in the taxonomy. To make the reasoning problem feasible we decompose it into three weakly connected subproblems and generate the probabilistic knowledge bases for each substep independently to keep the knowledge bases as small as possible: (1) inferring the relevant PRAC, (2) disambiguation and role assignment and (3) inferring missing information pieces and refinements of action cores and their associated roles.

Reasoning in PRAC about the most probable interpretation of a natural-language instruction ι is implemented by the following multi-step composition of database transformations by means of probabilistic relational inference:

$$\arg\max_{R_{A_{missing}}} P\left(R_{A_{missing}} \middle| \arg\max_{R_{A_{given}}} P\left(R_{A_{given}} \middle| \arg\max_A P\left(A \middle| I\right)\right)\right),$$

where $I = \mathscr{I}(\iota)$ is a PRAC instruction representing the syntactic structure of the NL instruction, A is the action core referred to by the instruction, $R_{A_{given}}$ are the action role assignments of A given in I, and $R_{A_{missing}}$ are the action roles of A which do not have a correspondence in I. Figure 6 depicts the reasoning pipeline of PRAC, which we will describe in the following in more detail.

1. Parsing: The first step in PRAC reasoning is to analyze the syntactic structure of the instruction at hand, which yields a PRAC instruction database I according to (1) containing syntactic relations and the part of speech for each word.
2. Given the words and their part of speech, the possible word meanings are obtained from the PRAC dictionary and the actioncore is identified that is 'activated' by I with highest probability:

$$\widehat{A} = \arg\max P\left(action\_core(a, ac) \;\middle|\; I, \mu(I) \right)$$

3. Given $\widehat{A}$ and its associated roles $\widehat{R}$ from the actioncore library, PRAC performs simultaneous word sense disambiguation and action role assignment taking into account the concept taxonomy of the PRAC dictionary:

$$\widehat{A}' = \arg\max P\left(has\_sense(\cdot,\cdot), \widehat{R} \;\middle|\; \widehat{A} \cup I \cup \sqsubseteq \right)$$

4. Subsequently, having assigned the action roles for the identified actioncore, PRAC checks if there is a plan schema in the plan library attached to the actioncore, which can be parameterized with the inferred roles. If so, the schema is instantiated with its parameters and sent to the plan executive.
5. If there is no plan schema attached, PRAC incrementally computes refinements of $\widehat{A}'$ by alternately solving for the most probable actioncore ac' *ac* can be AchievedBy and its action roles:

$$\widehat{A}'' = \arg\max_{a'} P\left(achievedBy(a, a') \;\middle|\; \widehat{A}' \cup \sqsubseteq \right)$$

A visualization of an exemplary inference process in PRAC and the execution of an instantiated plan schema can be found in the video accompanying this paper[3].

4 Related Work

In recent years, much work has been done in order to make knowledge sources available to robots, which are indented for human use [27, 24], and to generate robot plans out of natural-language instructions [16, 25, 10, 20, 27]. Dzifcak *et al.* [10] use a combinatorial categorial grammar for deriving a goal formulation in temporal logics in order to find an action sequence that achieves this goal. Matuszek *et al.* [16] use statistical machine translation techniques to match natural-language navigation directives against a formal path description language. Others [25, 24] use probabilistic models to derive plans to be executed by a robot. Misra *et al.* [18] take into account the context of the environment for grounding objects in an instruction to objects in the environment. They solve the ambiguity in instructions using an energy function corresponding to a conditional random field. What these approaches have in common is that they do not take into account that natural-language instructions typically are severely underspecified, ambiguous and often not directly executable. They make what is commonly referred to as the *closed-world assumption* postulating that all knowledge about the world is given and complete. Additionally, most approaches to teach robots by means of natural language are designed to capture and execute what is specified by an instruction using 'shallow' mappings to robot control, but they are not intended to accumulate more semantic action knowledge that can be recalled in and adapted to different situations. Artzi *et al.* [2] and Kim *et al.* [15] learn probabilistic context-free grammars for robot navigation tasks. Their

[3] https://youtu.be/iA6s7IGqubs

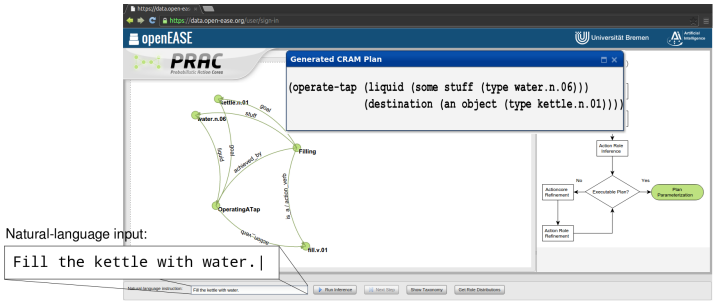

Fig. 7 The browser-based webinterface to PRAC on the openEASE cloud robotics platform.

approach is inspired from a more linguistic point of view, where such grammars are are typically induced from large corpora of text consisting of sequences of navigational directives.

We take a different approach accounting for the variational complexity and richness of human-scale manipulation tasks with everyday objects. In PRAC, the result of a linguistic analysis of an instruction is taken only as evidence in a probabilistic first-order knowledge base which allows us on the one hand to include any syntactic characteristics of a sentence as evidence in a query, and on the other hand it enables tight integration with the robot's belief state, high-level knowledge base, executive and perception system, which can provide comprehensive context information, such as the objects perceived in a scene, for instance. In addition, PRAC makes use of a rich taxonomy of concepts which allows to transfer the lernt knowledge to new, unseen concepts. Our work is not about finding action sequences given a particular goal, but about *how* to perform complex everyday activities in presence of partial and incomplete information. It is inspired by and closely related to Minsky's [17] frame representation and partially adapted from the FrameNet [4] specifications of action verbs but extended and adapted for including knowledge necessary for robot action execution. Our ultimate goal is a complete robotic agent that is able to successfully *perform* complex manipulation tasks formulated in NL (cmp. [1, 5]). Commonly used linguistic corpora of instructions do not account for the behavior that the analyzed instruction produces. This makes a large-scale corpus-based evaluation of PRAC nearly impossible. Our future work therefore focuses on thoroughly evaluating PRAC with respect to these points, such as the executability of an action or whether or not it produces the desired effects and avoids undesired effects by executing the generated robot plans in a simulated environment (cmp. [11]).

5 Conclusions

In this paper we have shown how interpretation of ambiguous and underdetermined natural-language instructions can be formulated as the problem of computing the

most probable complete and unique instruction in an action specific knowledge base called probabilistic action core (PRAC). Within the PRAC framework, the most probable complete and unique instruction enables robots to find the most appropriate plan and with the most general refinement of the formal plan parameters given the NL instruction. To perform this inference the PRAC framework learns a joint probability distribution over all possible ways in which instructions for a given action verb can be formulated. Our PRAC framework provides an attractive alternative to other instruction interpretation approaches, in particular for the interpretation of complex manipulation tasks. One important advantage is that PRACs are not limited to inferring which sequences of actions should be executed but also *how* the individual actions are to be executed. A second advantage is that the use of taxonomic reasoning in the PRAC inference results in the inference of the most general concept refinements of the plan parameters. This generates least commitment calls of plans that keep maximal flexibility at execution time and avoids the necessarity of grounding symbolic names that are generated in the interpretation process (symbol grounding problem). Our current implementation comprises a set of 12 PRACs and plan schemata from two application domains, the household/cooking domain and the domain of conducting chemical experiments, which we are continuously extending. We implemented PRAC as an open-source software framework which is accessible as a web service on the cloud robotics platform openEASE [7] (http://www.open-ease.org), shown in Figure 7.

Acknowledgements This work is supported by the EU FP7 Projects *RoboHow* (grant number 288533) and *ACAT* (grant number 600578).

References

1. J. Anderson. *Constraint-directed Improvisation for Everyday Activities*. PhD thesis, 1995.
2. Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.
3. D. Bailey. *When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs*. PhD thesis, UNIVERSITY of CALIFORNIA, 1997.
4. C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.
5. R. Barker. *Ecological psychology: Concepts and methods for studying the environment of human behavior*. Stanford Univ Pr, 1968.
6. M. Beetz, D. Jain, L. Mösenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic. Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE*, 100(8):2454–2471, 2012.
7. 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. Finalist for the Best Conference Paper Award and Best Cognitive Robotics Paper Award.
8. M. De Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

9. M.-C. de Marneffe and C. D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
10. J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4163–4168. IEEE, 2009.
11. J. Feldman and S. Narayanan. Embodied meaning in a neural theory of language. *Brain and Language*, 89(2):385 – 392, 2004. Language and MotorIntegration.
12. C. Fellbaum. *WordNet: an electronic lexical database*. MIT Press USA, 1998.
13. J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. Technical report 672, Yale University, Department of Computer Science, January 1989.
14. M. Georgeff and F. Ingrand. Decision making in an embedded reasing system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 972–978, Detroit, MI, 1989.
15. J. Kim and R. J. Mooney. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444. Association for Computational Linguistics, 2012.
16. C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 251–258. ACM, 2010.
17. M. Minsky. A framework for representing knowledge. Technical Report Memo 306, MIT-AI Laboratory, 1974.
18. D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
19. L. Mösenlechner and M. Beetz. Parameterizing Actions to have the Appropriate Effects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 25–30 2011.
20. E. Neo, T. Sakaguchi, and K. Yokoi. A natural language instruction system for humanoid robots integrating situated speech recognition, visual recognition and on-line whole-body motion generation. In *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, pages 1176–1182. IEEE, 2008.
21. 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.
22. D. Nyga and M. Beetz. Reasoning about Unmodelled Concepts – Incorporating Class Taxonomies in Probabilistic Relational Models. In *Arxiv.org*, 2015. Preprint: http://arxiv.org/abs/1504.05411.
23. M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006.
24. J. Ryu, Y. Jung, K. Kim, and S. Myaeng. Automatic extraction of human activity knowledge from method-describing web articles. *Proceedings of the 1st Workshop on Automated Knowledge Base Construction*, page 16, 2010.
25. S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. Natl Conf. on Artificial Intelligence (AAAI)*, 2011.
26. J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.
27. 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)*, pages 1486–1491, Anchorage, AK, USA, May 3–8 2010.