Intel Serv Robotics (2017) 10:159–171
DOI 10.1007/s11370-017-0216-y

ORIGINAL RESEARCH PAPER

A light non-monotonic knowledge-base for service robots

Luis A. Pineda[1] · **Arturo Rodríguez**[1] · **Gibran Fuentes**[1] · **Caleb Rascón**[2] · **Ivan Meza**[1]

Received: 7 July 2015 / Accepted: 10 January 2017 / Published online: 1 February 2017

Abstract In this paper a Non-Monotonic Knowledge-Base (KB) for practical applications in service robots is presented. The KB is defined as a conceptual hierarchy with inheritance that supports the expression of defaults and exceptions. All classes and individuals, with their properties and relations, can be updated dynamically and the KB-System supports non-monotonic behavior. Non-monotonicity is handled on the basis of a specificity criteria, such that more specific properties and relations have precedence over more general ones. The system supports the expression of conceptual (or terminological) and factual (or assertional) knowledge, which are used in inference in a coherent and consistent way. The KB-System is embedded within the IOCA Architecture, where knowledge about how to communicate and interact with the world, and also knowledge of the particular interpretation situation are represented. The cognitive architecture is structured around a main communication cycle, and queries and conceptual inferences are performed on demand during the interaction of the robot with other agents or the world. The overall structure of the KB with its main interpreter and supporting utilities as well as the embedding of the KB-system in the robot's architecture are also presented. The KB-System is illustrated with a case study in service robots scenarios, where a practical non-monotonic KB is required. Finally, the implementation of the KB-System in the robot Golem-III is described.

Keywords Knowledge representation in service robots · Non-monotonic KB-systems · The Golem-III robot

✉ Luis A. Pineda
lpineda@unam.mx

[1] Department of Computer Science, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad Universitaria, C.P. 04510 Coyoacán, México, D. F., Mexico

[2] Consejo Nacional de Ciencia y Tecnología (CONACyT), Commissioned to: Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad Universitaria, C.P. 04510 Coyoacán, México, D. F., Mexico

1 Knowledge representation in service robots

1.1 Conceptual hierarchies and terminological knowledge

Service robots need to deploy different kinds of knowledge and perform inferences in order to assist people in daily life chores successfully. One kind is conceptual knowledge, like natural taxonomies, which is commonly expressed through conceptual or class hierarchies. This knowledge structure consists of a partition of the set of objects in the universe of discourse (i.e., the top class) into a number of mutually exclusive subsets or more specific classes, which in turn can be partitioned into a further more specific classes and so on. Individuals of classes can have a number of properties and stand in a number of relations with individuals of the same or other classes. For instance, *eagles are birds*, *penguins are birds*, *birds are animals* and *fish are animals* are branches of a class hierarchy, and *fly* and *eat* are the names of a property of birds and a relation standing between eagles and fish, respectively (i.e., birds fly and eagles eat fish). In this knowledge structure, subordinated classes inherit the properties and relations of their superordinates classes, allowing conceptual inferences, like eagles are animals, eagles fly and some animals eat fish.

Conceptual hierarchies may be defined in terms of general concepts or classes only, but they can also include specific individuals, with their particular properties and relations; for

instance, it can be stated that Pete is a large eagle and Arthur a clever penguin, and that Pete and Arthur are not friends. In this case, properties and relations are determinate and hold only for the individuals involved. Individuals stand in a set membership relation with the classes they belong to, and, in addition to their particular properties and relations, inherit all properties and relations of their superordinate classes.

Conceptual hierarchies may represent complete or incomplete knowledge, depending on the interpretation conventions. The case of complete knowledge is exemplified by the so-called Closed-World Assumption, used in Prolog and many other representational systems; under this assumption the known facts and relations about the domain are represented in the KB, and if something is not represented explicitly it is simply because it is false. Under this interpretation, for instance, the question of whether fish eat animals in relation to the taxonomy above would be answered *no*; although this assumption is useful in many settings, it is nevertheless a limitation of the expressive power of the representation system which can yield false inferences, as in the present case. However, if the taxonomy is interpreted under the assumption that the knowledge expressed is incomplete, the answer to the same question would be that the system does not know, which would be the correct answer. Conversely, the expression of incomplete knowledge requires the capability to express that something is false; for this the `not` operator is required, and a system with such expressive power should be able to answer both *no* and *I don't know*.

The statement of properties and relations of the conceptual classes in a taxonomy corresponds implicitly to the expression of a set of definitions, and a taxonomy corresponds to a dictionary with possibly more than one entry for every class; for this reason this kind of knowledge is often referred to as "terminological". For instance, the taxonomy above grants the definitions *eagles are carnivorous birds* or *eagles are birds that eat fish*, among a set, possibly large, of definitions for all classes, including alternative less specific definitions for the same class.

Terminological knowledge is very stable and is shared by a linguistic community as reflected in common dictionaries, and this body of knowledge consists of the things that one knows or can conclude from the knowledge of the language only or in relation to a given conceptual structure. Terminological knowledge resembles also the kind of information stored in semantic networks (e.g. [2]) and more generally in semantic memory [45]. For all these reasons, this kind of knowledge is kept in a modular structure in the KB, which has been called the T-Box since the KL-ONE system [5], and the distinction is kept to the date in description logics [3].

Conceptual hierarchies with classes and instances are commonly expressed in object-based programming languages like Java, C++ and related paradigms, that support inheritance, but these approaches adopt the closed-world assumption implicitly and cannot express incomplete information, depend on the interpretation strategy and have a very limited inferential power; in addition, the KB needs to be compiled in advance and cannot be updated easily. For these reasons KB-Systems that overcome these limitations are required for supporting inference in intelligent service robots.

1.2 Non-monotonic inference and the principle of specificity

Although conceptual knowledge is very stable and readily accessible on demand, the expression of defaults and exceptions make conceptual inferences defeasible, and hence its representational system needs to be non-monotonic [40]. For instance, using Reiter's classic example [33], if we learn that penguins do not fly, and state such negative property in the conceptual hierarchy, the question of whether birds fly should be answered *yes* but whether penguins fly should be answered *no*, despite that penguins are birds. The non-monotonicity of the taxonomy translates into the terminological knowledge, and dictionary entries become contradictory, and dictionaries become inconsistent. For instance, the entry *birds are animals that fly* is inconsistent with the entry *penguins are birds that do not fly*.

In the present paper, we introduce a simple and intuitive mechanism which is capable of dealing with non-monotonicity in conceptual hierarchies expressing incomplete knowledge. This mechanism is based on the principle of specificity as follows: in case of conflict between the contents of a knowledge structure, more specific knowledge has precedence or supersedes less specific or more general one [40]. In the current example, birds fly but penguins do not, and as penguins is a more specific class than birds, its negative property has precedence in the interpretation, and the question of whether penguins fly is answered *no*. As can be seen the more general or less specific property is the default and the more specific one is the exception; however, defaults can have a negative nature and exceptions a positive one. For instance, if we extend our example with the classes mammals and platypus, and state the properties that mammals do not lay eggs but platypus do, the default is the negative and the exception the positive. We also consider that particular individuals are more specific to the classes they belong to (i.e., the set membership relation is more specific than the subset relation), and the principle of specificity also holds for particular individuals with their properties and relations. So, following with the example above Pete, the eagle, flies, but Arthur, the penguin, does not.

The principle of specificity is illustrated with the conceptual hierarchy of our current example in Fig. 1, where the defaults and exceptions with both positive and negative character can be appreciated directly. In this diagram, classes are

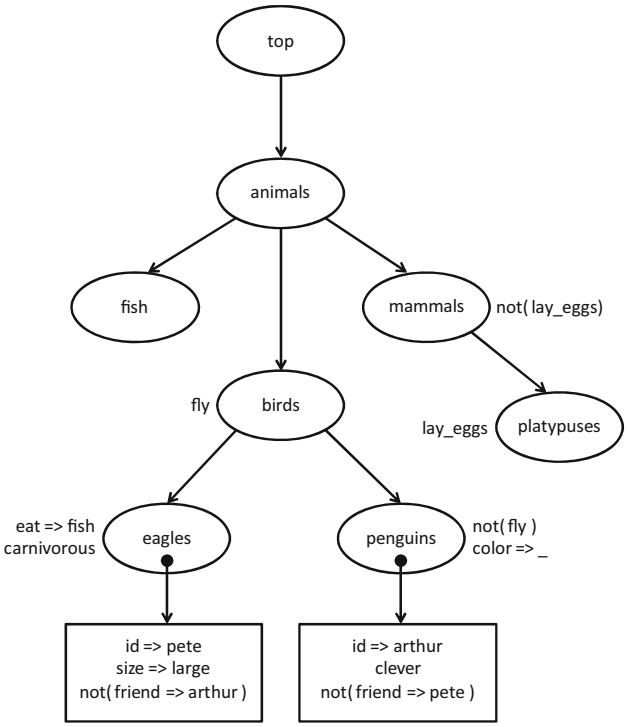

Fig. 1 Class hierarchy with defaults and exceptions

represented by ovals and particular instances by dots within their most specific classes pointing to squares holding the information of the corresponding specific objects. The symbol => relates a property or a relation of the corresponding class or individual with its value (e.g., eagles eat fish, size of Pete is large, Pete is not a friend of Arthur), where the value can be underspecified, like the color of penguins which is left undetermined.

In the present KB-System, the principle of specificity is used under the presupposition that the conceptual hierarchy expresses incomplete knowledge, and the KB can be modified by extending, changing or eliminating classes, properties and relations dynamically, and the interpretation renders, nevertheless, the right conceptual inferences. The principle of specificity is also independent of the interpretation strategy, and both top-down and bottom-up interpreters of the conceptual hierarchy that use this principle can be built, as discussed in Sect. 3.2.

The representation of non-monotonic knowledge and performing non-monotonic inference are very important problems in AI, and there is a very large body of literature on the subject, since the so-called Truth-Maintenance Systems or TMS [10], including the tradition of default and non-monotonic logics started by Reiter [33] and, more recently, Answer Set Programming [6], but the application of this body of work to service robots is still limited, as discussed in Sect. 2. Object-based programming languages and related paradigms use an implicit specificity criteria to resolve defaults and exceptions too, but these approaches have strong limitations for representation and inference, as discussed above.

1.3 Factual or assertional knowledge

Service robots need also represent factual knowledge about specific scenes in the world, consisting of a set of concrete individuals, with their properties and relations, including states, events and processes, which may extend through space and time. This kind of knowledge is often referred to as assertional, alluding to the kind of speech acts through which statements about the world are expressed and interpreted in spoken conversation.

For instance, Pete in front of Arthur, who is sitting on an ice cube, is a concrete scene or state of the world at a particular point in time. The scene can undergo events, like the penguin standing up, or processes, like the penguin running and the eagle chasing it, and the representation of these events and processes constitutes factual knowledge too. If we also learn that Arthur throw a stone to Pete and broke his wing, so Arthur fell to the ground, the knowledge of such action, event and process, with their causal and temporal relations, constitute assertional knowledge as well. Although the present example may suggest that factual knowledge corresponds to knowledge acquired by visual perception, the notion is much more general and corresponds to observational or empirical knowledge acquired by the different modalities of perception or other sensory devices, and consists of the facts of arbitrary problems in arbitrary domains.

Factual knowledge is commonly involved in deliberative inferences, like diagnosis, decision making, planning and, more generally, in problem solving. These kinds of inferences are carried on by the application of valid inference schemes upon factual information and can be construed as a symbolic search process on a problem space, whose states are representations of potential states of the world. For all these reasons, factual or assertional knowledge is commonly stored in an independent memory module which is often referred to as the assertional box or A-Box, specially in the tradition of description logics. Terminological and factual knowledge are of course related in representation and inference, and the symbols in the A-Box need to point to or be associated with the corresponding symbols in the T-Box.

In the KB-system presented in this paper, we take advantage of the capability of the conceptual hierarchy to express concrete individuals with their corresponding properties and relations for all classes, in conjunction with the specificity of set membership (between individuals and classes) and the subset relation (between classes), for the expression of assertional knowledge, as illustrated by the eagle and the penguin story. Spatial scenes can be represented explicitly by stating the concrete spatial properties and relations of concrete

objects in relation to the robot. In this way, the demarcation between the T-Box and the A-Box is a matter of the degree of specificity from the abstract to the concrete. Also, as the concrete objects are more specific than their classes, their particular properties and relations become exceptions that take precedence over defaults, and the specificity principle handles factual and terminological knowledge in a simple and coherent way.

1.4 The interaction-oriented cognitive architecture

The KB-system presented in this paper supports the functionality described above and is implemented as a module or service of the IOCA cognitive architecture, that we have developed in the context of the Golem project [30,31]. This architecture is illustrated in Fig. 2.

The architecture has three main functional levels, that we call reactive, perceptual and representational or deliberative from the bottom to the top. The bottom level involves the recognition devices (e.g., the visual object recognition system, the speech recognition system, etc.), the autonomous reactive agent for each modality, and the rendering devices which are also modality specific. The middle level involves the perceptual interpretation processes on the input side, and the specification of the external actions on the output. The top-level is constituted by the interpreter of the SitLog programming language [32] for the definition of the dialogue models, through which the context, interaction and communication knowledge is expressed; this language is also used to define and interpret the services of the KB-System as well as the deliberative inferential resources, like diagnosis, decision making and planning systems, which are used opportunistically during the main flow of the communication cycle, although the details of these latter systems are beyond the scope of the present paper.

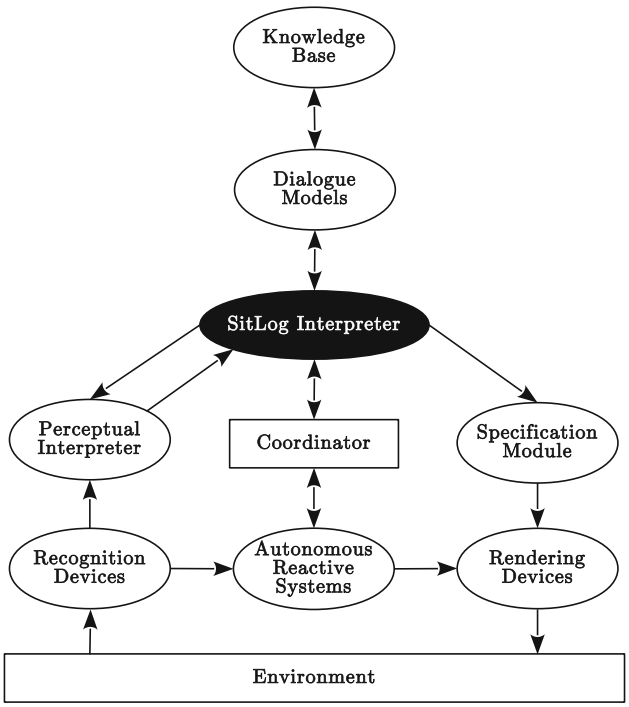

Fig. 2 KB and the interaction-oriented cognitive architecture

2 Related work

In this section, we summarize some related work on knowledge bases for service robots; see [21] for a thorough review on the topic. Knowledge in service robots can be acquired either manually before the execution of the task, dynamically through interaction with people and the environment or by making inferences from the available knowledge and the current context. In service robotics there has been a considerable amount of work on knowledge acquisition from the interaction with humans (e.g. through natural language [20,23,24,35], multimedia interaction [7,23]) and the robot's environment (e.g. locations [20,34], people's location [19,41], objects [1]). Another important body of research has been devoted to finding ways of integrating declarative domain knowledge and probabilistic reasoning (e.g. [36,46]). The use of fuzzy knowledge bases have been also widely investigated in other areas of robotics such as navigation and control [15,17,28,29,38].

To the best of our knowledge, most of the proposed systems for knowledge representation and reasoning in service robots are monotonic, including logic oriented systems, and, in particular Description Logic (e.g. [12,22,43]). Other approaches use a combination of monotonic logic for conceptual inferences although allow non-monotonic behavior in deliberative inference, for instance, for spatial reasoning (e.g. [25]). Among the most popular instances of these is KnowRob [42,43], which is written in Prolog and uses the Web Ontology Language (OWL) (based on Description Logic). The KnowRob system allows the specification of different knowledge types, includes various inference schemes and knowledge acquisition and grounding methods and provides close integration with control programs. KnowRob has been widely used in different complex service robot tasks such as helping set a breakfast table [27], manipulating tools [4], performing actions cooperatively [11] and semantic mapping [44]. Another popular instance is the OpenRobots Ontology [22], written in Java and also built on the OWL, which has been used for different tasks such as object learning [18] and interactive object manipulation [39].

For non-monotonic systems, on its part, Answer Set Programming (ASP) (e.g. [8,47]) is perhaps the most popular approach. A representative instance of its use in non-monotonic knowledge representation and reasoning is Ke

Jia's robot task planning system [8,9] built on the Causal Calculator [14], which implements a type of non-monotonic causal logic (a form of ASP). The task planning system has also been equipped with a mechanism to take advantage of open-knowledge [9] and has been demonstrated in the RoboCup@Home competition where it is coupled with natural language processing, motion planning and robot control. In semantic mapping, the frame-based language LOOM [26], which is capable of non-monotonic reasoning, has been used for representing semantic knowledge and integrating it with spatial knowledge [13]. The semantic maps produced by this system have been further exploited for task planning in service robots applications. Some non-monotonic OWL-DL (a sublanguage of OWL) reasoners have been also integrated in service robots systems and exploited for learning an unknown environment after an incomplete tour [16] and for learning to make decisions from human demonstrations [37].

The approach described in this paper represents knowledge by maintaining an explicit ontology and supporting conceptual inferences with clear regimented classes, individuals, properties and relations, resembling description logics; however, as opposed to most of these approaches, the present KB-system enables non-monotonic reasoning via a simple and efficient computational mechanism based on the principle of specificity, integrates the expression of complete and incomplete knowledge in a simple and transparent way, and supports questions and updates of both terminological and assertional or factual knowledge along the execution of the task and the interaction with the user in real-time, offering a very good compromise between efficiency and expressiveness for KB-Systems in service robots.

3 Structure and functionality of the KB-system

3.1 Structure of the knowledge-base

In this section we present the structure and interpretation of the KB and survey its supporting utilities. The main KB object is the class such that every class is represented through a standard Prolog's predicate with six arguments, as follows:

```
class(Name, Mother, Props, Rels,
                Instances)
```

where `Name` and `Mother` are the identifiers (i.e., constants or atoms) of the corresponding objects, and `Props`, `Rels` and `Instances` are lists, possibly empty, with one entry for each property, relation or instance of the class. There is a most general class (i.e., the top) whose mother is none and the list of descendants of the more specific classes is empty. The KB itself is a list of classes and all types of objects (i.e., classes, properties, relations and instances) are independent of the order in which they are declared, and although they are represented as Prolog lists, they are interpreted as bags. Hence, all these objects can be easily consulted and updated at any level of granularity, and there is a set of utilities to handle all list operations on the KB. The full Prolog code for all the KB-Services is available at http://golem.iimas.unam.mx/lightkb.

Properties and relations are named with positive or negative atoms (e.g., `constant` and `not(constant)` respectively), and terms are of the form `atom`, `attribute => value` or `not(attribute => value)`, where `value` is a positive atom or a list of atoms, predicates or attribute value structures, and the operator => states a relation between an attribute and its value, where the value can also be underspecified, as discussed above in relation to Fig. 1.

Each individual is represented through a list with three elements: the individual's name or list of names, the list of its specific or concrete properties and the list of its concrete relations with other individuals (i.e., [`Id`, `Props`, `Rels`]), where `Id` is an attribute value pair of the form `id => Name` and `Name` is the actual name of the corresponding individual or a list with the alternative names of such an individual. This value can also be underspecified and the system supports the representation of anonymous individuals.

In this notation the properties and relations are specified in relation to the class or the individual having them, and the => operator is interpreted within the scope of the corresponding class or individual, so `eat => fish` within the scope of the class of eagles is interpreted as eagles eat fish, and `size => large` and `not(friend => arthur)` within the scope of Pete are interpreted as Pete is large and Pete is not a friend of Arthur respectively. The notation also permits the underspecification of the object of relations, when these are indeterminate; for instance, the atom `eat` can be included in the list of relations of eagles, and interpreted as stating that all eagles stand in an eating relation with individuals of an unspecified class, although the relation can also be reified as a property, and included in the corresponding list, meaning that eagles have the property of eating. For instance, the class hierarchy depicted in Fig. 1 is represented in Listing 1.

```
class(top, none, [], [], []).
class(animals, top, [], [], []).
class(birds, animals, [fly], [], []).
class(fish, animals, [], [],[]).
class(mammals, animals, [not(lay-eggs)], [], []).
class(eagles, birds, [carnivorous], [eat=>fish], [[id=>pete, [size=>large],
                                                   [not(friend=>arthur)]]]).
class(penguin, birds, [not(fly), color=>_], [], [[id=>arthur, [clever], [not(friend=>pete)]]]).
class(platypus, mammals, [lay-eggs], [], []).
```

Listing 1: Representation of Terminological and Factual Knowledge

The conceptual hierarchy can be extended for the explicit representation of states, processes, actions carried on by intentional agentes, and natural events occurring in the world. These conceptual objects can be included as classes in the taxonomy. Classes have the same structure as before and are interpreted by the same KB-Services exactly. The only extension is that we add a compulsory attribute in the list of properties of particular instances of these classes, whose value is a list of grounded predicates with the actual description of the particular state, process, action or event, from the point of view of the external observer (i.e., `description => List_of_Predicates`). The list of relations of the instances of these classes, on its part, includes only temporal and causal relations between states, processes and events that can be used for modeling temporal and causal inferences, although the description of such functionality is beyond the scope of the present paper. These additions are illustrated with the descriptions of the states, processes, actions and events of the eagle and the penguin story in Listing 2.

3.2 Basic KB-system services and conceptual inferences

Conceptual inferences are supported by six main services provided by the KB, as follows:

1. *class_extension*: it provides the extension of a referent class under the closure of the inheritance relation. In particular, `class-extension(top, Extension)` returns the whole set of individuals in the KB.
2. *property_extension*: it provides the extension of a referent property under the closure of the inheritance relation; this is, `property-extension(property, Extension)` returns the whole set of individuals that have such property in the KB.
3. *relation_extension*: it provides the extension of a referent relation under the closure of the inheritance relation. This is, `relation-extension(relation, Extension)` returns the whole set of individuals that stand as subjects in such relation in the KB.

```
class(top, none, [], [], []).
class(states, top, [], [], [[id=>s1, [description=>[in_front(pete, arthur),
                                                    has_broken_wing(pete),
                            sitting_on(arthur, 'ice cube')]],
                             [after=>p2]]]).
class(processes, top, [], [], [[id=>p1, [description=>[flying(pete)]], []],
                  [id=>p2, [description=>[falling(pete)]], []],
                  [id=>p3, [description=>[chasing(pete, arthur), running(pete),
                                                    running(arthur), behind(pete, arthur)]],
                                       [after=>a2]]]).
class(actions, top, [], [], [[id=>a1, [description=>[throws(arthur, stone)]],
                                  [during=>p1, cause=>e1]],
                               [id=>a2, [description=>[stands(arthur)]],
                                       [after=>s1]]]).
class(events, top, [], [], [[id=>e1,[description=>[hit(stone,wing),part-of(wing,pete)]],
                                    [after=>a1,cause=>p2]]]).
```

Listing 2: Representation of States, Processes, Actions and Events

4. *classes_of*: it provides the whole set of classes including the referent class or individual, under the closure of the inheritance relation.
5. *properties_of*: it provides the whole set of properties that the referent class or individual has under the closure of the inheritance relation.
6. *relations_of*: it provides the whole set of relations in which the referent class or individual stands as subject under the closure of the inheritance relation.

These services are implemented as Prolog predicates with one input and one output arguments: the reference class, property or relation, and the list with the corresponding extension respectively or the corresponding set of classes, properties or relations. With this set of services it is possible to compute the set of classes or individuals, properties and relations that satisfy an arbitrary description or condition, which can be programmed as standard Prolog's predicates directly, and defined as a user function within the SitLog's environment. As was mentioned, the KB is fully declarative and all these services have a top-down and a bottom-up procedure which produce exactly the same result, and can be used indistinctly by the user or the application, although the bottom-up is usually more efficient.

The principle of specificity is implemented by building a list of classes, properties or relations of the referenced individual instances making sure that more specific objects are placed in front of the list, and when the KB is queried the object closest to the beginning of the list is selected. In the top-down procedure, the less specific objects are placed at the end of the list along the inspection of the conceptual hierarchy, and, conversely, in the bottom-up procedure, the more specific objects are placed at the front of the list during the traversal of the tree. Although a property or a relation can have more than one value within the list, possibly different, the selected value will be the right one at the current state of the KB. The specification of the representational structure and the full code in Prolog of both the top-down and the bottom-up interpreters and supporting utilities, which are also fully coded in Prolog, are also available at http://golem.iimas.unam.mx/lightkb.

Standard queries involving conceptual inferences are implemented with the six predicates in conjunction with standard Prolog programs, taking into account that the KB expresses incomplete information. These programs need to consider that both properties or their negation can be included in the KB, and also that queries can be phrased in positive and negative terms; so, querying whether eagles fly and penguins do not fly should be answered *yes*, and querying whether penguins fly and eagles do not fly should be answered *no*.

The procedure takes into account whether the queried property is positive or negative. For the positive case, it needs to be verified that the property is included in the list of properties of the class, or that its negation is included in such list, so the answer can be affirmative or negative in the strong sense. Conversely, if the question is phrased in negative terms, it has to be checked out that such negative property is in the class, so the answer is *yes*, or that the non-negated property is in the class, and the answer is *no*. If neither the property or its negation, for both positive or negative atoms, are included in the list, or the list of properties of the class is empty, the information is incomplete and the system answers *unknown*. For instance, if the question is whether penguins are carnivorous the answer is that the system does not know. The Prolog predicates `has_property` and `has_relation` that check the corresponding condition are given directly in Listing 3. The predicate `incomplete_information` verifies the four possible conditions, or whether the list of properties or relations is empty, and `check_front_atom` inspects the list from the front to the back to verify whether the atom (positive or negative) is included, and fails otherwise. In addition to the services available for classes, similar services are defined for asking whether a concrete individual has a property or stands in a concrete relation with another individual.

3.3 Multiple extensions and preferences

Non-monotonic reasoning systems face also the problem that default rules that can be potentially contradictory. For instance, suppose that it is asserted in the KB that the place of birth of penguins is the same as its predators, the eagles, and, at the same time, that penguins live where they were born. Then, if it is asserted that the birth place of Pete, the eagle, is Patagonia, then it follows that Patagonia is also where Arthur, the penguin, lives. However, suppose that Arthur is captured and moved to San Diego's Zoo, and this information is asserted in the KB. Then, it follows that Arthur lives both in Patagonia and in San Diego's Zoo. Suppose further that it is also stated that predators live where their preys live. Then, it would follow that Pete lives in San Diego's Zoo too! Furthermore, suppose that the constraint that the place where a particular individual lives must be only one is somehow added in the KB. Hence, the KB becomes inconsistent.

The present KB-system can be easily extended to express inferential default rules but the principle of specificity cannot handle these kinds of scenarios appropriately when conflicting defaults are expressed, and rendering the right result in a particular inference involving this kind of situation is a very hard logical problem.

```
has_property(Class, Property, Answer) :-
            properties_of(Class, Properties),
            incomplete_information(Property, Properties, Answer).

has_relation(Class, Relation, Answer) :-
            relations_of(Class, Relations),
            incomplete_information(Relation, Relations, Answer).

% The class has no properties or relations
incomplete_information(Atom, [], unknown).

% The atom is positive or negative
incomplete_information(Atom, List, yes) :- check_front_atom(Atom, List).

% The atom is negative
incomplete_information(not(Atom), List, no) :- check_front_atom(Atom, List).

% The atom is positive
incomplete_information(Atom, List, no) :- check_front_atom(not(Atom), List).

% The list contains neither the atom nor its negation
incomplete_information(_, _ , unknown).
```

Listing 3: Interpretation of the KB with Incomplete Information

However, this kind of rules are much weaker than standard defaults, like *all birds fly*, that are based on ontological facts, and for that reason can be handled as preferences rather than as properties and relations. For instance, a list of *preferences*, including such weaker defaults with an associated weight, in addition to the lists or properties and relations of classes and individuals, can be included in the structure of the KB, such that the defaults with the highest weight are placed at the front in the extension of properties and relations lists, of classes and individuals, at each particular state of the KB. However, such functionality is left for further research.

3.4 Conceptual hierarchies versus lattice or multi-hierarchies

The KB-System described in this section supports a strict hierarchy of classes, but at the same time has the expressive power corresponding to a lattice structure or a multi-hierarchy although with a much simpler and transparent semantics and a very reasonable computational cost. This is achieved by adopting a modeling constraint such that classes are defined solely on the basis of ontological criteria, a particular interpretation perspective and a naming convention, as opposed to sets or "classes" defined on the basis of a particular property or relation that all individuals of the set have necessarily.

For instance, if the set of individuals "having-lungs" were introduced in our example in Fig. 1 as a "class" dominated by *top* and dominating *mammals* and *birds* (i.e., a sister of "animals"), the structure would be a lattice instead of a hierarchy. Then if it is asserted in such KB that animals cannot sing while the members of class "having-lungs" can, then birds and mammals would inherit such artistic property and its negation, and the system would become inconsistent for this additional reason.

Reasoning about this kind of structures requires to enrich the inferential machinery with a preference mechanism to resolve the inconsistencies for this additional reason, with the corresponding increase in the complexity of the semantics and the computational cost. Otherwise, there is no way to guarantee that the system is consistent with large potential interpretation and computational problems.

However, in the present framework properties and relations "hang" from classes in the whole of the hierarchy, and the same property or relation can hold of any class without structural implications. For instance, following up with our example, it can be stated directly in the taxonomy in Fig. 1 that animals do sing as a default but fish do not as an exception, preserving the strict hierarchy with a simple and clear interpretation. For this, the present approach shows a very good compromise between its structural properties and expressive power on the one hand, and its transparent semantics and computational properties on the other.

Nevertheless, it has to be taken into account that a comprehensive view of the world may involve partitioning the domain in different forms at the same time, like parallel copies of the universe of discourse, each defining a particular conceptual hierarchy, and representing and reasoning about such knowledge in an integrated and systematic way would indeed require a lattice structure, but such study is beyond the scope of the present paper.

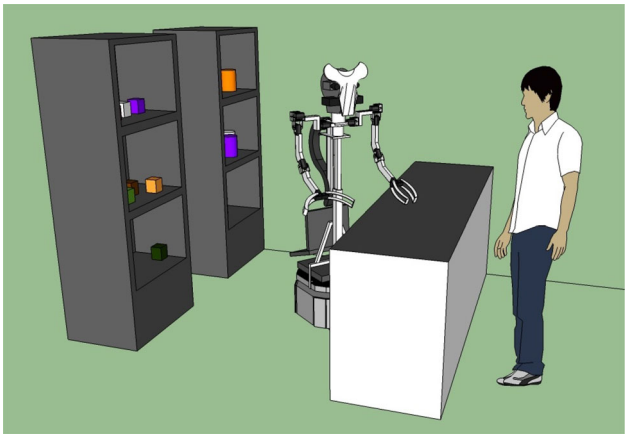

Fig. 3 A diagram of how the environment is setup

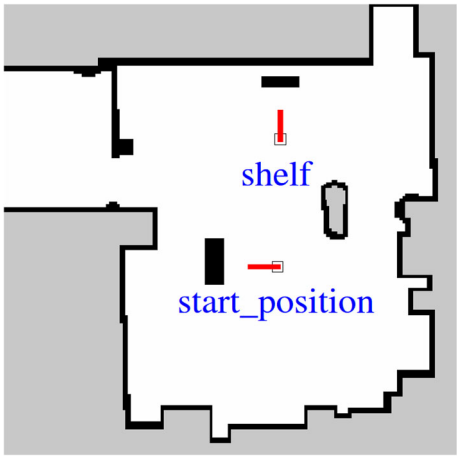

Fig. 4 The semantic mapping of the scenario

4 The KB-system in the robot Golem-III

4.1 An application of service robots using the KB

To demonstrate the applicability of the proposed KB, we have implemented and tested its full functionality in our service robot Golem-III. For this the scenario shown in Fig. 3, where the robot acts as a shop clerk that brings the objects a user asks for from their respective shelves, was setup.

To carry out the demonstration, semantic mapping is required. For the purpose of navigation, two locations are labeled and their coordinates are stored in the knowledge base: (1) a starting position, where the customer interaction takes place; and (2) a shelf, where all the products are stored and can be manipulated by Golem-III. The semantic mapping and the code used to describe it are shown in Fig. 4 and Listing 4, respectively. It is important to note that we defined a location class for each location (one for storage and another for human interaction) to be able to add more locations in each class in a future version of this application, since the KB is able to provide such flexibility. In addition, the starting position has an additional *height* property that can be used so that Golem-III delivers objects at an appropriate height.

The KB that Golem-III has at the start of the demonstration and its corresponding code are shown in Fig. 5 and Listing 5, respectively. The top node is the same shown in Listing 4 as the two pieces of code are parts of the same KB, so it is specified only once. This class hierarchy is similar to the one presented in Fig. 1, as to show how such structure can be useful in a typical service robotics task. A video showing the full demonstration as well as the full source code can be accessed at http://golem.iimas.unam.mx/lightkb.

```
% Defining the class tree, and the location class
class(top,none,[],[],[]),
class(entity, top, [], [], []),
class(location, entity, [], [], []),

% Defining a class of locations for storage
% and in it, the shelf location
class(storage, location, [], [], [[id => shelf,[coordinate=>[0,1.30,0]],[]]]),

% Defining a class of locations for human interaction
% and in it, the starting_position location
class(interaction, location, [], [], [
    [id => starting_position,[coordinate=>[0.5,0,0],height=>80.0],[]],
])
```

Listing 4: Code for the Semantic Mapping for the Scenario.

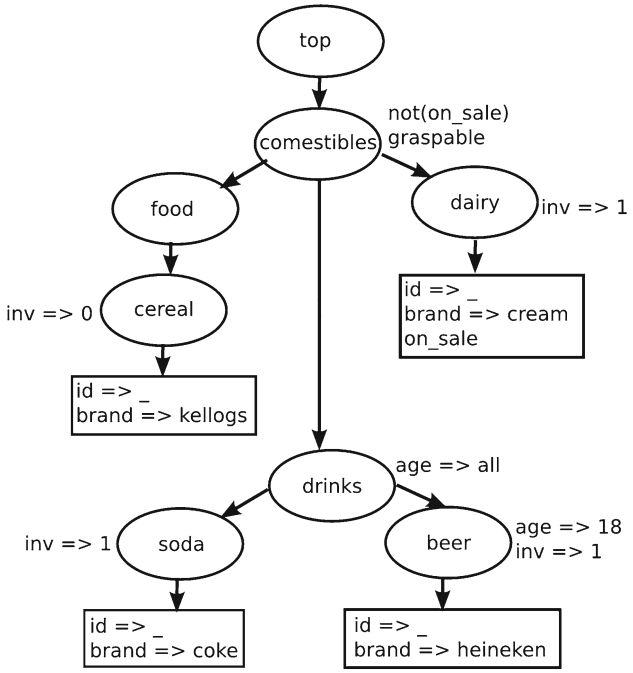

Fig. 5 The knowledge base used in the demonstration

This demo was presented life as the Open Challenge test of the Golem team in the @Home competition at RoboCup Leipzig 2016[1], obtaining the second highest score in such test.

In the start of the demonstration, there is an initial interaction where the user asks for several objects:

- **Cereal**: It is not in inventory, so Golem-III cannot bring it.
- **Toilet paper**: It is not in the KB, so Golem-III responds that it does not have enough information about it.
- **Beer**: In this case is Heineken, but it has the restriction that it can only be sold to 18-year-olds or older, which the user is not.
- **An non-alcoholic beverage**: In this case it is a coke, which Golem can deliver.

Once Golem-III reaches the shelf to grab the coke, it also recognizes that: (1) there is cereal in the inventory, and (2) the cream is tipped over, making it non-graspable, and updates these facts in the KB. Thus, once Golem-III comes back to the user, it announces that there is cereal in the inventory and asks if the user wants the robot to bring it.

The user at a later point also asks for a diary product that is on sale. All comestibles are not on sale, which is a negative default, but cream is an exception. The *graspable* property on its part is a default for all objects under the comestible tree, however, when an object is observed as not being graspable (as is the case of the cream in the scenario described), the *not(graspable)* property can be asserted of such particular object, and, thus, provide a much fuller description of its inventory.

The *not* operator, used to state negative defaults like *not(on_sale)* or negative properties of specific objects, is interpreted in the strong sense: that the referred objects do not have the corresponding property or do not stand in a negated relation. This contrast with the weak sense of the negation in which the system replies *no* whenever it does not really know. For instance, when being asked if there was any toilet paper, an object not in the KB, the answer based on the close-world assumption would be that there is not, which may be false in case a human clerk would have added it into the inventory without Golem-III being notified. In this case, because of the nature of the KB, Golem-III is able to provide a more grounded answer: it does not have enough information.

```
%Defining the comestible class
class(comestible, top,[graspable,not(on_sale)],[],[]),

%Defining the food branch
class(food, comestible,[],[],[]),
class(cereal, food,[inv=>0],[],[[id => c1,[brand=>kellogs],[]]]),

%Defining the drinks branch
class(drink, comestible, [age=>all],[],[]),
class(soda, drink,[inv=>1],[],[[id => s1,[brand=>coke],[]]]),
class(beer, drink,[age=>18,inv=>1],[],[[id => b1,[brand=>heineken],[]]]),

%Defining the dairy branch
class(dairy, comestible,[inv=>1],[],[[id => d1,[brand=>cream,on_sale],[]]])
```

Listing 5: Code describing the object class hierarchy from Figure 5.

[1] http://www.robocup2016.org/en/

As it can be seen, the flexibility the KB provides is an important asset for the description of the scenario, in both the objects in the inventory as well as its surroundings. The properties shown here (of both locations and objects) were only an example of what can be stored in the KB, of which there is no limitation. In fact, everything relevant to the task cannot only be stored in the KB, but can be accessed and modified as the task is being carried out. This accounts for an important accomplishment in the interaction with the human.

4.2 The robot Golem-III

The robot Golem-III is shown in Fig. 6. Its lower half, which houses the hardware used for navigational purposes, is an Adept MobileRobot's Research PatrolBot robotic base. Its upper half, which houses the rest of its hardware, is an in-house-built robotic torso. In Table 1, a brief summary is presented of its hardware and the software libraries used.

Table 1 Software libraries and hardware used by the Golem-III modules

Module	Hardware	Software libraries
Dialogue manager	–	SitLog
Vision	Microsoft Kinect 2, Point Grey Flea Camera	MOPED, Kinect for Windows SDK
Navigation	Adept MobileRobot Research PatrolBot	OpenSLAM, AMCL, Trajectory Rollout, Dynamic Window
Speech recognition	RODE VideoMic	Windows Speech API
Speech synthesis	Infinity 3.5-Inch Two-Way Speaker	Windows Speech API
Robot audition	8SoundsUSB	JACK
Object manipulation	In-house Built Robotic Torso, Arms and Grippers	Dynamixel RoboPlus
Camera/Mic. movement	In-house Built Robotic Neck	Dynamixel RoboPlus

5 Discussion

In this paper a light KB-System for service robots, with its associated inferential machinery has been presented. Terminological and factual knowledge is represented through strict class hierarchies with their corresponding individuals, but a propositional or logical representation is used for expressing properties and relations, with a very good compromise between expressivity and effective computation. Like first order logic and description logics, the present formalism permits the expression of incomplete information and supports the expression of strong and weak negation, and like default logics permits the expression of defaults and exceptions, in a coherent, natural and simple way. Also, like conceptual hierarchies and semantic networks, the present KB-System supports conceptual inference through the graph connectivity very efficiently. However, instead of using conceptual lattices or multi-hierarchies, that need to add preferences to handle inconsistencies due to inheritance through different paths, the present approach transfers such expressive burden to the propositional representation of properties and relations of all classes and individuals. For this, the present approach exhibits a very good trade-off between propositional and graphical formalisms for representing knowledge.

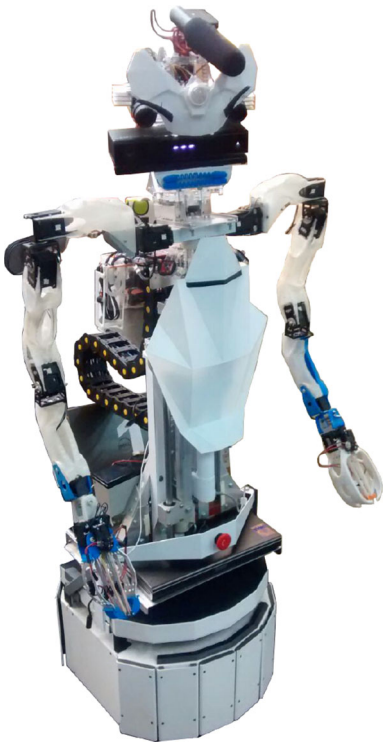

Fig. 6 The Golem-III service robot

Service robots require a cognitive architecture that uses different kinds of knowledge. The present approach, implemented in the IOCA architecture, has a main communication cycle characterized by speech act protocols that use opportunistically the terminological and factual knowledge. These protocols, specified with the SitLog programming language, define an explicit task structure, which in turn consists on the situations that the robots visits along the interaction with the user. Perceptual and action algorithms enabling the robot behavior are contextualized in relation to situations, simplifying greatly the interpretation and grounding of external information. Conceptual knowledge and inference are best thought of as modular resources that can be used on demand according to the context. In particular, ontologies representing knowledge of particular domains are powerful resources to support the robot's flexible behavior.

Acknowledgements We thank the support of Mauricio Reyes, Hernando Ortega, Noé Hernández, Ricardo Cruz, Varinia Estrada and the members of the Golem Group who participated in the development of the robot Golem-III. We also acknowledge the support of Grants CONACYT's 178673, ICYTDF-209/12 and PAPIIT-UNAM's IN-107513 and IN-109816.

References

1. Ahlrichs U, Fischer J, Denzler J, Drexler C, Niemann H, Noth E, Paulus D (1999) Knowledge based image and speech analysis for service robots. In: Proceedings of the integration of speech and image understanding, pp 21–47
2. Anderson JR, Bower GH (1980) Human associative memory: a brief edition. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey
3. Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF (2010) The description logic handbook: theory, implementation, and applications. Cambridge University Press, Cambridge
4. Becker J, Bersch C, Pangercic D, Pitzer B, Rühr T, Sankaran B, Sturm J, Stachniss C, Beetz M, Burgard W (2011) The pr2 workshop-mobile manipulation of kitchen containers. In: IROS workshop on results, challenges and lessons learned in advancing robots with a common platform, vol 120
5. Brachman RJ, Schmolze JG (1985) An overview of the kl-one knowledge representation system. Cognit Sci 9(2):171–216
6. Brewka G, Eiter T, Truszczyński M (2011) Answer set programming at a glance. Commun ACM 54(12):92–103
7. Burghart C, Mikut R, Stiefelhagen R, Asfour T, Holzapfel H, Steinhaus P, Dillmann R (2005) A cognitive architecture for a humanoid robot: a first approach. In: Proceedings of the IEEE-RAS international conference on humanoid robots, pp 357–362
8. Chen X, Ji J, Jiang J, Jin G, Wang F, Xie J (2010) Developing high-level cognitive functions for service robots. In: Proceedings of the international conference on autonomous agents and multiagent systems, vol 1, pp 989–996
9. Chen X, Lu D, Chen K, Chen Y, Wang N (2014) KeJia : the intelligent service robot for RoboCup@Home 2014. Tech. Rep., Multi-Agent Systems Lab., Department of Computer Science and Technology, University of Science and Technology of China
10. Doyle J (1979) Artificial intelligence. Truth Maint Syst 12(3):251–272
11. Fan Z, Tosello E, Palmia M, Pagello E (2014) In: Proceedings of the international conference intelligent autonomous systems
12. Galindo C, Saffiotti A, Coradeschi S, Buschka P, Fernandez-Madrigal JA, Gonzalez J (2005) Multi-hierarchical semantic maps for mobile robotics. In: Proccedings of the IEEE/RSJ international conference on intelligent robots and systems, pp 2278–2283
13. Galindo C, andez Madrigal JAF, alez JG, Saffiotti A (2008) Robot task planning using semantic maps. Robot Auton Syst 56(11):955–966
14. Giunchiglia E, Lee J, Lifschitz V, McCain N, Turner H (2004) Nonmonotonic causal theories. Artif Intell 153(1–2):49–104
15. Hagras HA (2004) A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. IEEE Trans Fuzzy Syst 12(4):524–539
16. Hawes N, Hanheide M, Hargreaves J, Page B, Zender H, Jensfelt P (2011) Home alone: autonomous extension and correction of spatial representations. In: robotics and automation (ICRA), 2011 IEEE international conference on, pp 3907–3914
17. Hoffmann F, Pfister G (1997) Evolutionary design of a fuzzy knowledge base for a mobile robot. Int J Approx Reason 17(4):447–469
18. Ivaldi S, Nguyen SM, Lyubova N, Droniou A, Padois V, Filliat D, Oudeyer PY, Sigaud O (2014) Object learning through active exploration. IEEE Trans Auton Ment Dev 6(1):56–72
19. Karg M, Kirsch A (2012) Acquisition and use of transferable, spatio-temporal plan representations for human-robot interaction. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, pp 5220–5226
20. Kollar T, Perera V, Nardi D, Veloso M (2013) Learning environmental knowledge from task-based human-robot dialog. In: Proceedings of the IEEE international conference on robotics and automation, pp 4304–4309
21. Lemaignan S (2012) Grounding the interaction: knowledge management for interactive robots. Ph.D. thesis, Université de Toulouse
22. Lemaignan S, Ros R, Msenlechner L, Alami R, Beetz M (2010) Oro, a knowledge management platform for cognitive architectures in robotics. In: Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on, pp 3548–3553
23. Lemaignan S, Ros R, Alami R, Beetz M (2011) What are you talking about? grounding dialogue in a perspective-aware robotic architecture. In: Proceedings of the IEEE RO-MAN, pp 107–112
24. Lemaignan S, Ros R, Sisbot EA, Alami R, Beetz M (2012) Grounding the interaction: anchoring situated discourse in everyday human-robot interaction. Int J Soc Robot 4(2):181–199
25. Lim GH, Suh IH, Suh H (2011) Ontology-based unified robot knowledge for service robots in indoor environments. IEEE Trans Syst Man Cybern Part A Syst Hum 41(3):492–509
26. MacGregor R, Burstein MH (1991) Using a description classifier to enhance knowledge representation. IEEE Expert 6(3):41–46
27. Pangercic D, Tenorth M, Jain D, Beetz M (2010) Combining perception and knowledge processing for everyday manipulation. In: IEEE/RSJ international conference on intelligent robots and systems, pp 1065–1071
28. Petković D, Issab M, Pavlović ND, Zentnerb L, Žarko Ćojbašić (2012) Adaptive neuro fuzzy controller for adaptive compliant robotic gripper. Expert Syst Appl 39(18):13,295–13,304
29. Petković D, Shamshirband S, Anuar NB, Sabri AQM, Rahman ZBA, Pavlović ND (2016) Input displacement neuro-fuzzy control and object recognition by compliant multi-fingered passively adaptive robotic gripper. J Intell Robot Syst 82:177–187
30. Pineda LA, Meza I, Aviles H, Gershenson C, Rascon C, Alvarado M, Salinas L (2011) IOCA: interaction-oriented cognitive architecture. Res Comput Sci 54:273–284
31. Pineda LA, Rodríguez A, Fuentes G, Rascon C, Meza IV (2015) Concept and functional structure of a service robot. Int J Adv Robot Syst 12:6. doi:10.5772/60026
32. Pineda LA, Salinas L, Meza IV, Rascon C, Fuentes G (2013) SitLog: a programming language for service robot tasks. Int J Adv Robot Syst 10:358. doi:10.5772/56906
33. Reiter R (1980) A logic for default reasoning. Artif Intell 13:81–132
34. Schiffer S, Ferrein A, Lakemeyer G (2012) Reasoning with qualitative positional information for domestic domains in the situation calculus. J Intell Robot Syst Theory Appl 66(1–2):273–300
35. Schiffer S, Hoppe N, Lakemeyer G (2013) Natural language interpretation for an interactive service robot in domestic domains. In: Filipe J, Fred A (eds) Agents and artificial intelligence, communications in computer and information science, vol 358. Springer, Berlin Heidelberg, pp 39–53
36. Schmidt-Rohr SR, Knoop S, Lösch M (2008) Bridging the gap of abstraction for probabilistic decision making on a multi-modal service robot. In: robotics: science and systems IV, pp 105–110
37. Schmidt-Rohr SR, Dirschl G, Meissner P, Dillmann R (2011) A knowledge base for learning probabilistic decision making from human demonstrations by a multimodal service robot. In: Proceedings of the international conference on advanced robotics, pp 235–240

38. Seraji H, Howard A (2002) Behavior-based robot navigation on challenging terrain: a fuzzy logic approach. IEEE Trans Robot Autom 18(3):308–321
39. Sisbot EA, Ros R, Alami R (2011) Situation assessment for human-robot interactive object manipulation. In: RO-MAN, pp 15–20
40. Strasser C, Antonelli GA (2014) Non-monotonic logic. In: Zalta Ward N (ed) The stanford encyclopedia of philosophy (Winter 2014 Edition). Academic Press, New York
41. Stückler J, Behnke S (2011) Improving people awareness of service robots by semantic scene knowledge. In: Ruiz-del Solar J, Chown E, Plöger PG (eds) RoboCup 2010. Springer, Heidelberg, pp 157–168
42. Tenorth M, Beetz M (2013) Knowrob: a knowledge processing infrastructure for cognition-enabled robots. Int J Robot Res 32(5):566–590
43. Tenorth M, Beetz M (2015) Representations for robot knowledge in the KnowRob framework. Artif Intell. doi:10.1016/j.artint.2015.05.010
44. Tenorth M, Kunze L, Jain D, Beetz M (2010) Knowrob-map - knowledge-linked semantic object maps. In: IEEE-RAS international conference on humanoid robots, pp 430–435
45. Tulving E (2013) Memory systems: episodic and semantic memory. In: Tulving E, Donaldson W (eds) Organization of memory. Academic Press, New York, pp 381–403
46. Zhang S, Sridharan M, Gelfond M, Wyatt J (2012) KR3: An architecture for knowledge representation and reasoning in robotics. In: Proceedings of the international workshop on non-monotonic reasoning
47. Zhang S, Sridharan M, Sheng Bao F (2012) ASP+POMDP: integrating non-monotonic logic programming and probabilistic planning on robots. In: Proceedings of the IEEE international conference on development and learning and epigenetic robotics