Draft Version
Final version published in Robotics and Autonomous Systems, 2013

Inferring Robot Goals from Semantic Knowledge

Cipriano Galindo*

Department of System Engineering and Automation
University of Málaga, Campus de Teatinos
Málaga, Spain.

Alessandro Saffiotti

AASS Cognitive Robotic Systems Lab
Örebro University, Sweden

Abstract

A growing body of literature shows that endowing a mobile robot with semantic knowledge, and with the ability to reason from this knowledge, can greatly increase its capabilities. In this paper, we present a novel use of semantic knowledge: to encode information about how things should be, or *norms*, and allow the robot to infer deviations from these norms and to generate goals to correct these deviations. For instance, if a robot has semantic knowledge that perishable items must be kept in a refrigerator, and it observes a bottle of milk on a table, this robot will generate the goal to bring that bottle into a refrigerator. The key move is to encode norms in an ontology, so that each norm violation results in a detectable inconsistency. A goal is then generated to bring the world back in a consistent state, and a planner is used to transform this goal into actions. Our approach provides a mobile robot with a limited form of *goal autonomy*: the ability to derive its own goals to pursue generic aims. We illustrate our approach in a full mobile robot system that integrates a semantic map, a knowledge representation and reasoning system, a task planner, as well as standard perception and navigation routines.

Keywords:
Semantic maps, Mobile robotics, Goal autonomy, Proactivity, Norms, Ontology, Description logics, Inconsistency, Spatial representation, Conceptual maps.

1. Introduction

Mobile robots intended for service and personal use are being increasingly endowed with the ability to represent and use semantic knowledge about the environment where they operate [1]. This knowledge encodes general information about the entities in the world and their relations, for instance: that a kitchen is a type of room which typically contains a refrigerator, a stove and a sink; that milk is a type of perishable food; and that perishable food is stored in a refrigerator. Once this knowledge is available to a robot, there are many ways in which it can be exploited to better understand the environment or plan actions [2, 3, 4, 5, 6], assuming of course that this knowledge is a

*Corresponding author: cipriano@ctima.uma.es

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

faithful representation of the properties of the environment. There is, however, an interesting issue which has received less attention so far: what happens if this knowledge turns out to be in conflict with the robot's observations?

Suppose for concreteness that the robot observes a milk bottle laying on a table. This observation conflicts with the semantic knowledge that milk is stored in a refrigerator. The robot has three options to resolve this contradiction: (a) to update its semantic knowledge base, e.g., by creating a new subclass of milk that is not perishable; (b) to question the validity of its perceptions, e.g., by looking for clues that may indicate that the observed object is not a milk bottle; or (c) to modify the environment, e.g., by bringing the milk into a refrigerator. While some work have addressed the first two options [7, 5, 8], the last one has not received much attention so far. Interestingly, the last option leverages an unique capability of robots: the ability to modify the physical environment. The goal of this paper is to investigate this option.

We propose a framework in which a mobile robot can exploit semantic knowledge to identify inconsistencies between the observed state of the environment and a set of general, declarative descriptions, or *norms*, and to generate *goals* to modify the state of the environment in such a way that these inconsistencies would disappear. When given to a planner, these goals lead to action plans that can be executed by the robot. This framework can be seen as a way to enable a robot to proactively generate new goals, based on the overall principle of maintaining the world consistent with the given declarative knowledge. In this light, our framework contributes to the robot's *goal autonomy*.

Our framework relies on a *hybrid semantic map*, which combines semantic knowledge based on description logics [9] with traditional robot maps [7]. Incoherences between the sensed reality and the model, i.e., the observation of facts that violate a particular norm, will lead to the generation of the corresponding goal that, when planned and executed, will re-align the reality to the model, as in the milk bottle example discussed above. It should be emphasized that in this work we only focus on the goal inference mechanism: the development of the required sensorial system, and the possible use of semantic knowledge in that context, are beyond the scope of this paper.

The rest of this paper is organized as follows. In the next section we review some related work. Section 3 introduces our semantic map. Section 4 presents our approach to encode norms as semantic knowledge, to detect norm violations, and to generate goals to correct these violations. Section 5 extends our approach to the case of multiple concurrent norm violations. In section 6, we report a proof-of-concept experiment that shows the concrete applicability of our approach to real robotic systems. Finally some conclusions and future work are outlined.[1]

2. Related work

The fact that future robots will have to be endowed with semantic knowledge is being increasingly recognized by the robotics community [1, 11, 12]. Most current approaches rely on a shallow interpretation of semantic knowledge: the data used by the robot are simply augmented with labels, like "door" or "kitchen", which carry a semantic meaning to humans, but this meaning is not explicitly represented into the robot. Often these semantic labels are used for human-robot interaction [13, 14]. Many

[1]The work reported here is a significant extension of the one presented earlier at the European Conference on Mobile Robotics 2011 [10].

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

proposals have been made to allow the robot to acquire these labels automatically [15, 2, 4, 16, 17], even in a life-long perspective [18].

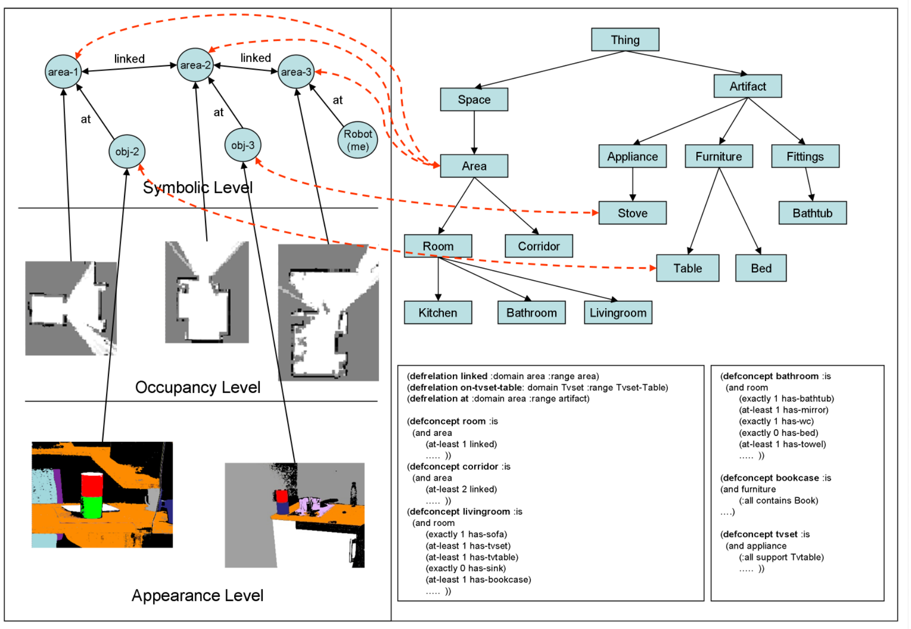
A few proposals exist, however, that take a deep semantic stance, in that semantic labels are embedded in a domain theory and are put in relation with other categories in some form of ontology. A robot can then effectively use this deep semantic knowledge for reasoning. For example, a robot may include an ontology that represents the relation that a kitchen is a type of room which contains a stove. This robot could use the fact that a room is labeled as "kitchen" to form the expectation that there is a stove in it; conversely, if the robot detects a stove in a room it will classify that room as a kitchen [5]. Proposals that adhere to a deep semantic stance include [3, 5, 6, 7, 8, 19, 20, 21]. The European project RoboEarth goes one step further, and use ontologies not only to allow a robot to perform new inferences, but also to enable meaningful communication among heterogeneous robots [22].

The above approaches have shown that endowing a robot with an explicit representation of semantic knowledge can increase the robot's behavioral autonomy, by improving their basic skills (planning, navigation, localization, etc.) with deduction abilities. However, few researchers have addressed the use of semantic knowledge to increase the robot's *goal* autonomy, that is, the robot's ability to generate its own goals given generic motivations. Two notable exceptions are the CuriousGeorge project [3] and the CogX project [21]. In both cases, the authors explore the ability of the robot to generate its own perceptual goals, based on some innate "curiosity" that pushes it to increase its knowledge. In our work, by contrast, the robot generates its own *action* goals, based on some innate "sense of order" that pushes it to maintain its environment in good order with respect to a given set of norms, encoded in a declarative way in its internal semantic representation.

The concept of goal autonomy of a robot has been previously discussed in connection with the more general concept of autonomy. Robot autonomy is most typically understood as the ability of a physical agent to accomplish tasks without the intervention of external agents [23]. Beside this interpretation, sometimes called *behavioral autonomy*, another type of autonomy is sometimes considered, called *goal autonomy*: the robot's ability to generate its own goals (tasks), that is, the pro-activeness of the agent [24, 25]. While behavioral autonomy has been widely addressed in the robotic arena by developing deliberative architectures and robust algorithms for planning and executing tasks under uncertainty, goal autonomy has received less attention in robotics, being mostly explored in the fields of multi-agent systems [26, 27, 28] and implemented through motivational architectures [29, 30].

Our approach to goal autonomy can be seen as a case of normative goals applied to agents which act based on beliefs and intentions [26, 27]. However, normative goals are often considered as simple *if-then* rules triggered when particular stimuli are given in the environment [31, 29]. Other works have used the term *maintenance goals* to represent innate goals that are aimed to satisfy a particular state of the world over time, e.g., the battery level should be always over a certain value [32, 33]. Our approach substantially diverges from those works, since it is not based on procedural rules, i.e., motivation-action pairs, nor if-then rules. Instead, we rely on a declarative representation of the domain, from which the robot *infers* what should be done according to the current factual information in order to maintain the consistency between the environment and its representation.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013



```
(defrelation linked :domain area :range area)
(defrelation on-tvset-table: domain Tvset :range Tvset-Table)
(defrelation at :domain area :range artifact)

(defconcept room :is
  (and area
      (at-least 1 linked)
      ..... ))
(defconcept corridor :is
  (and area
      (at-least 2 linked)
      ..... ))
(defconcept livingroom :is
  (and room
      (exactly 1 has-sofa)
      (at-least 1 has-tvset)
      (at-least 1 has-tvtable)
      (exactly 0 has-sink)
      (at-least 1 has-bookcase)
      ..... ))
```

```
(defconcept bathroom :is
  (and room
      (exactly 1 has-bathtub)
      (at-least 1 has-mirror)
      (exactly 1 has-wc)
      (exactly 0 has-bed)
      (at-least 1 has-towel)
      ..... ))

(defconcept bookcase :is
(and furniture
      (:all contains Book)
....)

(defconcept tvset :is
  (and appliance
      (:all support Tvtable)
      ..... ))
```

Figure 1: An example of semantic map for a home-like environment. S-Box is on the left and T-Box on the right. See explanation in the text. Figure from [5].

3. A Semantic Map for Mobile Robot Operation

The semantic map considered in this work, derived from [5], comprises two different but tightly interconnected parts: a *spatial box*, or S-Box, and a *terminological box*, or T-Box. Roughly speaking, the S-Box contains factual knowledge about the state of the environment and of the objects inside it, while the T-Box contains general semantic knowledge about that domain, giving meaning to the entities in the spatial box in terms of concepts and relations. For instance, the S-Box may represent that `Obj-3` is placed at `Area-2`, while the T-Box may represent that `Obj-3` is a stove which is a type of appliance. By combining the two sides, the semantic map can infer that `Area-2` is a kitchen, since it contains a stove.

This structure is reminiscent of the structure of hybrid knowledge representation (KR) systems [9], which are dominant in the KR community. In these systems, the knowledge base consists of a terminological component, called T-Box, that contains the description of the relevant concepts in the domain and their relations; and an assertional component, called A-Box, storing concept instances and assertions about those instances. Our semantic map extends the assertional component to be more than a list of facts about individuals by also associating these individuals to sensor-level information with a spatial structure — hence the name S-Box. Similar approaches sometime go under the name of object maps [13, 18, 20].

Figure 1 shows a simple example of a semantic map of a home-like environment where both the S-Box and the T-Box have a hierarchical structure. The hierarchy in the T-Box is a direct consequence of the fact that the represented semantic knowledge forms a taxonomy. In our work, we use OWL-DL as a formalism to represent semantic knowledge [34]. OWL-DL is a widely used KR formalism, and many ontologies and

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

reasoning tools are available for it. For the S-Box, the use of a hierarchical spatial representation is a convenient and common choice in the robotic literature [35, 36, 37] for dealing efficiently with large-scale environments. Of course one could also consider a flat representation in the S-Box: in fact, in our framework, the S-Box can be substituted by any other spatial representation.

In the next section we exploit this semantic map, and in particular the knowledge and inference mechanisms of the T-Box, to infer robot goals.

4. Inferring Goals from Semantics

The semantic map described above provides two different points of view of the robot workspace. On the one hand the spatial part (S-box) enables the robot to generate plans from basic skills, striving for behavioral autonomy. On the other hand the terminological part (T-box) provides an abstract model of the robot environment which includes general knowledge, e.g., *books are located on shelves*, which can be exploited for the automatic generation of robot goals.

First we give an informal description of the proposed mechanism for goal generation. Then, section 4.2 formalizes our approach in description logic. Finally, section 4.3 illustrates the process with an intuitive example.

4.1. Informal Description

In the field of knowledge representation, semantic knowledge is usually interpreted as being *descriptive* of a specific domain: for example, the item of knowledge “beds are located in bedrooms” is used to partially describe beds. This knowledge is useful to infer implicit properties from a few observed facts. For example, if the robot perceives a bed in a room it can infer that the room is a bedroom; conversely, if it is commanded to find a bed it can restrict its search to bedrooms. Galindo *et al.* [5] offer examples of applications of these inferences in the robotic domain.

Interestingly, semantic knowledge can also be interpreted as being *normative*: under this interpretation, the above item of knowledge is prescribing where a bed must be located. The difference becomes apparent when considering how a robot should react to an observation that contradicts this knowledge. Consider again the milk box example in the Introduction, and the three possible options to resolve the contradiction discussed there. Options (a) (*update the model*) and (b) (*update the perceived state*) correspond to different ways to modify the robot’s beliefs to recover from a contradiction, and are related to execution monitoring and to uncertain state estimation. These options have been explored previously [7, 38]. The third option (c) (*update the physical world*) is more suited for contradictions that involve normative knowledge, and leads to goal generation. This is the option addressed here.

Informally, our approach defines a subset of concepts and relations stored in the T-Box as *normative*, i.e. they are involved in norms that should be fulfilled, by defining a special class `normative-concept` and a special relation `normative-relation`. Items of knowledge to be treated as normative will derive from them.

For instance, we can define that the normative concept `towel` should be related to the concept `bathroom` through the *normative* relation `place` — that is, towels *should* be located in a bathroom.

When a given instance violates a norm in the T-Box, the reasoning system derives the items of knowledge involved in the norm, and hence the goal that should be posted in order to change the state of the world in such a way that the norm is satisfied. In our

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

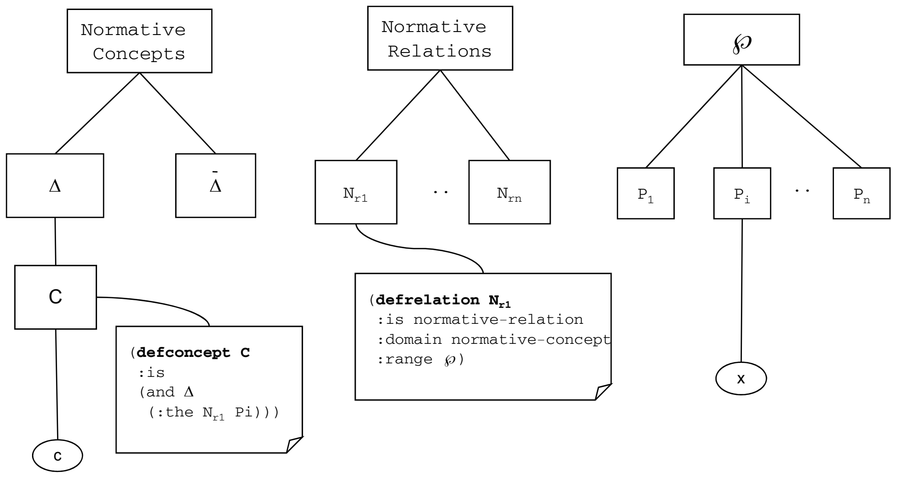


Figure 2: Knowledge representation for detecting inconsistencies. Boxes represent concepts while instances are represented as circles. The concept C is defined as a normative concept related to P_i through the normative relation N_{r1}. See explanation in the text.

example, suppose that an instance of a towel is perceived in a room which is not a bathroom. Then the given definition of a `towel` is violated — a circumstance that can be detected by most knowledge representation systems, including the Pellet reasoner [39] used in our experiments. Since the above definition of towel is normative, our approach yields a goal to satisfy the constraint, that is, to make the `place` of this towel be an instance of a `bathroom`. If the robot knows, e.g., that `room-3` is a bathroom, then the goal "bring the towel to `room-3`" is generated.

4.2. Description Logic Representation for Inferring Normative Goals

We now give a more precise characterization of our goal generation approach using the formalism of Description Logic – see, e.g., [40] for an introduction to this formalism. Let $\mathfrak{I}$ be a description logic interpretation on a particular domain $\mathcal{D}$. Let $\wp$ define a set of disjoint concepts $\wp = \{P_1, \dots P_n\}$, i.e., $\forall a, a \sqsubseteq P_i \Rightarrow \nexists j, j \neq i, a \sqsubseteq P_j$, where $x \sqsubseteq y$ denotes that x is subsumed by concept y.

Let N_r be called a *normative relation*, a function defined as:

$$N_r : N_C \to \wp$$

where N_C represents the set of the so-called *normative concepts*, that is, concepts which ought to be properly related to those from $\wp$. N_r actually defines the norm to be kept: we denote by N_R the set of all normative relations. Each normative relation is defined as one-to-one function: $\forall b \sqsubseteq N_C \Rightarrow \dot{\exists} P_j \in \wp, b \to [FILLS : N_r\ P_j]$, where $a \to [FILLS : B\ C]$ denotes that instance a is related through relation B to some instance derived from concept C.

The set N_C is further divided into two disjoint sets: the set $\triangle$ of all normative concepts that fulfill the imposed norms, and the set $\overline{\triangle}$ of those that fail to fulfill some of the norms (see figure 2).

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

Norm violation detection. Within this structure of the domain, norm violations are automatically detected when any instance is deduced to belong to multiple disjoint concepts. For example, let C be a normative concept (and therefore $C \sqsubseteq \triangle$ by definition) which is related to the P_i concept through the normative relation N_r. That is,

$$\forall c \sqsubseteq C, c \rightarrow [FILLS : N_r \; x], x \sqsubseteq P_i$$

If in a given interpretation $\mathfrak{I}$, $\exists k \sqsubseteq C, k \rightarrow [FILLS : N_r \; y], y \sqsubseteq P_j \in \wp, P_j \neq P_i$, then $\mathfrak{I} \models y \sqsubseteq P_j \wedge y \sqsubseteq P_i$, which implies that y is incoherent. In other words, if the normative relation N_r is not met for a particular instance k of a normative concept C, then the filler of that instance, in this case y, becomes incoherent. Moreover, since k is defined as $k \sqsubseteq C \sqsubseteq \triangle$, it is also inferred that $k \sqsubseteq \overline{\triangle}$, which also makes k incoherent.

Goal Inference. Given an incoherent instance k of a normative concept C, $k \sqsubseteq C$, and a normative relation $N_r, N_r(k) = x, x \sqsubseteq P_i \in \wp$, the inferred goal to recover the system from the incoherence is:

```
(exists ?z (P$_i$ ?z) (N$_r$ k ?z))
```

That is, we post the goal to be in a state in which there exists an instance z of P_i which is related to k through the normative relation N_r. Note that it is not necessary to include in the goal the requirement that k is not in relation N_r with any other $z' \neq z$, since the N_r function is defined one-to-one.

4.3. Implementation

To better understand the proposed approach, we describe a concrete example implemented using off-the-shelf systems for both knowledge representation and for planning. Knowledge is represented in OWL-DL [34] using the Protegé ontology editor [41] coupled with the Pellet inference system [39]. Planning is performed using JSHOP [42], a planner based on HTN, Hierarchical Task Networks [43]. These tools are very common in the respective domains, but other tools could have been used without major changes.[2] Our algorithms for norm violation detection and goal generation have been implemented in Java and interfaced with Pellet and JSHOP through their standard API.

The example is inspired by the one presented above, but does not involve physical perception and physical execution: the observed facts are manually introduced into Protegé, and the generated plan is only printed to the user. Section 6 will show an illustrative experiment in which our goal generation system is coupled to a real robotic system, providing physical perception and action.

Scenario. Consider a home assistant robot taking care of the apartment shown in Fig. 3. The apartment is composed of five rooms, i.e., a kitchen (`k1`), a bathroom (`b1`), a livingroom (`r1`), a bedroom (`r2`), and a home office (`r3`). A fragment of the concept ontology for this scenario is shown in Fig. 4. It models `Towel`, `Milk-Bottle`, and `Shoe` as normative concepts. It also models a set of disjoint concepts for rooms, and one for containers, i.e., places on/in which objects can be placed. The norms are encoded in the semantic definition of normative concepts, as shown in Fig. 5. These definitions impose that bottles of milk should be kept inside a fridge, towels must be placed in bathrooms and shoes in bedrooms.

[2]In a previous version of this work [10], the LOOM KR system and the PTLPlan planner were used.

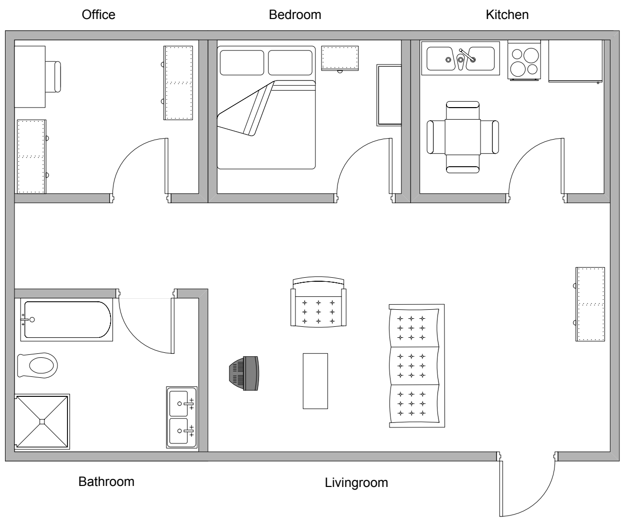


Figure 3: Apartment layout.

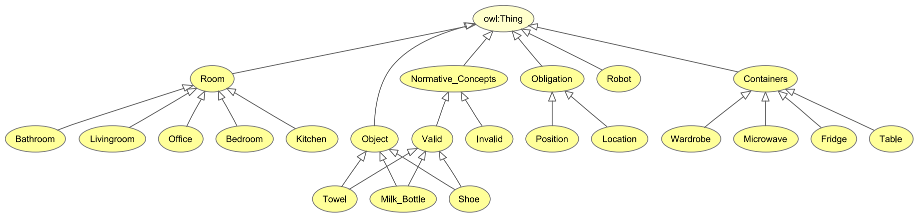


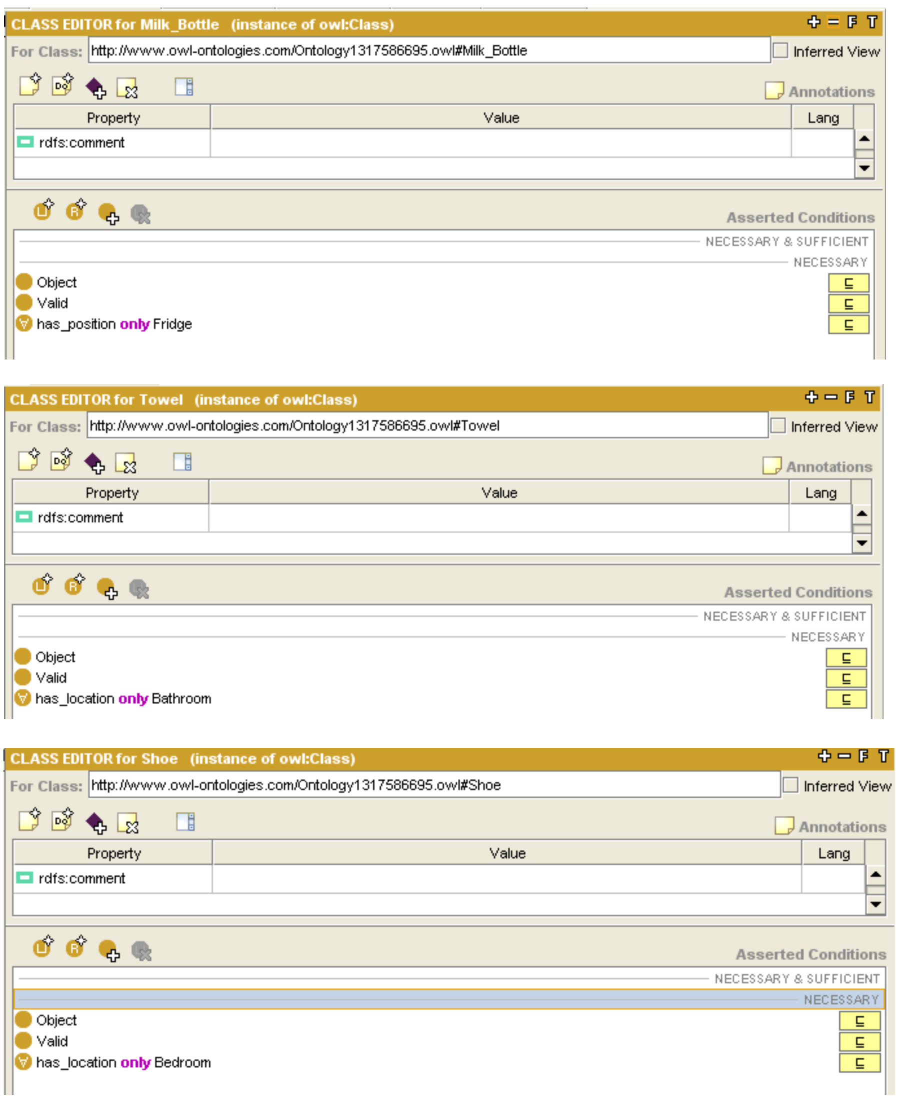
Figure 4: Fragment of concept ontology.

Norm violation detection. Let's assume that the robot is located at the livingroom `r1`, and that from there it observes a towel, called `t1`, inside the kitchen `k1`. The inclusion of this information into our KR system results in an inconsistency, which corresponds to the fact that the norm that towels must be in a bathroom has been violated.

More precisely, since towel `t1` is located in room `k1`, and towels are defined to be (normatively) located at bathrooms, then `k1` is deduced to be an instance of the concept `Bathroom`. However, `k1` is already asserted to be an instance of `Kitchen`, and bathrooms and kitchens are mutually exclusive concepts. This inconsistency is detected by Pellet, which reports it through the following message:

```
[disjointWith(Kitchen, Bathroom),
 subClassOf(Towel, all(has-location, Bathroom)),
 prop(has-location, t1, k1),
 type(t1, Towel),
 type(k1, Kitchen)]
```

Goal generation and planning. To resolve the inconsistency, our algorithm generates the goal to satisfy the violated norm, and passes this goal to the HTN planner to generate a sequence of actions which achieves it.



CLASS EDITOR for Milk_Bottle (instance of owl:Class)

For Class: http://www.owl-ontologies.com/Ontology1317586695.owl#Milk_Bottle

Property	Value	Lang
rdfs:comment		

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

Object ⊑
Valid ⊑
has_position **only** Fridge ⊑

CLASS EDITOR for Towel (instance of owl:Class)

For Class: http://www.owl-ontologies.com/Ontology1317586695.owl#Towel

Property	Value	Lang
rdfs:comment		

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

Object ⊑
Valid ⊑
has_location **only** Bathroom ⊑

CLASS EDITOR for Shoe (instance of owl:Class)

For Class: http://www.owl-ontologies.com/Ontology1317586695.owl#Shoe

Property	Value	Lang
rdfs:comment		

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

Object ⊑
Valid ⊑
has_location **only** Bedroom ⊑

Figure 5: Definition of the normative concepts. Norms are implicitly encoded in these definitions: for instance, a `Towel` is defined to have an unique valid location, which must be a `Bathroom`.

The computation of the goal starts from considering the abstract goal discussed in the previous subsection:

(exists ?z (P_i z) (N_r k ?z))

In our case, this becomes:

```
(exists ?z (Bathroom ?z) (has-location t1 ?z))
```

To convert this into a concrete goal for the HTN planner, our goal generation algorithm selects from the planning domain (see Appendix B) a method with a postcondition satisfying the violated norm. In our example, it searches a method that can result in

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

a state where `(has-location t1 ?z)` is true, with the variable `?z` bound to any instance of the range of the violated norm (here, `b1` since this is the only instance of `Bathroom`). Our algorithm selects the `bring` method, since it includes the `!drop` operator which has `has-location)` as a postcondition. This yields to the following generated goal:

```
(bring t1 b1)
```

If there are several possible instances of the range of the violated normative relation (here, if the apartment has several bathrooms), then a selection criteria is used: in our algorithm, we simply select the first instance returned by the KR query engine. Notice that backtracking may be needed in this case, if the selected instance yields a goal which is not satisfiable.

Finally, the above goal is given to the SHOP2 HTN planner, which, based on the planning domain in Appendix B, generates the following plan:

```
(!move r1 k1)
(!pickup t1 k1)
(!move k1 r1)
(!move r1 b1)
(!drop t1 b1)
```

5. Managing Multiple Norm Violations

The treatment above is adequate for dealing with the violation of a single norm, by transforming this violation in a detectable inconsistency and generating a goal that, when executed, will restore the norm. In this section we deal with cases in which multiple norms are violated in the same situation, and show how multiple simultaneous norm violations can be detected, prioritized, and restored.

5.1. Inferring multiple normative goals

The inference of multiple concurrent normative goals becomes not trivial when dealing with tableau reasoners, as it is the case of most of the commonly used reasoners. Recall that in our approach each norm violation causes the knowledge system to become inconsistent, given that the entity involved in the violation is inferred to be an instance of two mutually exclusive concepts. Unfortunately, once the knowledge system is inconsistent, the reasoner cannot make further inferences, and thus only one norm violation can be detected. Moreover, the specification of the goal, once the norm violation is detected, requires retrieving instances of general classes which is not possible if the knowledge system is inconsistent.

In order to deal with multiple norm violations, we have extended our approach in an iterative way using the following steps:

1. Query the KR the cause of the inconsistency, storing the normative relation violated, N_r, and the instances involved in the violation.
2. Remove the violated normative relation from the knowledge base; repeat step 1 while the KR system reports an inconsistency.
3. Once all the violated normative relations are removed, and thus the KR system is consistent again, the goal that solves each violation is computed as explained in section 4.2.
4. Restore the KR system with the previously removed norms.
5. Compute plans for the list of generated goals.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

5.2. Prioritized normative goals

An issue that arises from having multiple normative goals is how to prioritize them in order to make the robot to firstly attend to those norm violations which are most important for the considered domain. In this work we address this issue by assigning to each normative concept and to each normative relation a value that indicates their priority level. We do so through the functions $prio_c$ and $prio_r$:

$$prio_c : N_C \to [0, 1]$$

$$prio_r : N_R \to [0, 1]$$

The domain of $prio_c$ is naturally extended to individual instances by postulating:

$$prio_c(k) = \max_{C \in N_C : k \sqsubseteq C} prio_c(C),$$

that is, the priority of an individual k is the highest priority associated to any normative concept C of which k is an instance.

For each normative goal $g = (exists\ z\ (P_i\ z)\ (N_r\ k\ z))$, the assigned priority is then computed by

$$prio(g) = prio_r(N_r) * prio_c(k)$$

Once all goals have been ranked according their priorities, they could be given to a planner capable of finding the optimal subset of achievable goals, like [44] or [45]. In this paper, however, we chose the simpler strategy to generate and execute a separate plan for each goal. Goals with higher priority will be planned and executed before the ones with lower priority. For example, if the normative concept `Milk-Bottle` and to the normative relation `place` have been assigned a high priority, then the robot will accomplish first the task of bringing the milk into the fridge, as shown in the example below.

5.3. Example

To illustrate the case of multiple concurrent norm violations, we extend the example presented in Sec. 4.3 to the case in which a number of violations simultaneously occur.

Scenario. Imagine that our domestic cleanup robot, when exploring the apartment in Fig. 3 after a party, observes the following state of affairs:

- A towel (`t1`) is seen in the kitchen;
- Another towel (`t2`) is seen in the livingroom;
- A shoe (`o1`) is found in the kitchen;
- A bottle of milk (`o1`) is seen on the bedroom table;

this state contains four norm violations: two towels are not located in a bathroom, a shoe is not located in a bedroom, and a bottle of milk is outside the fridge.

The above observations are entered in the factual knowledge base of Protegé, resulting in the set of facts listed in Fig. 6. These also include facts known *a priori*, like the fact that there is a table in the bedroom, a fridge in the kitchen, plus information used for robot navigation, like the current position of the robot (`r2`) and connectivity information (the `nav` relation which is defined as symmetric).

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

```
[instances]
t1 Towel
t2 Towel
b1 Bathroom
k1 Kitchen
r1 Livingroom
r2 Bedroom
r3 Office
o1 Shoe
mb1 Milk_Bottle
f1 Fridge
table-1 Table
[facts]
has_location o1 k1
has_location robot-1 r2
has_location t1 k1
has_location t2 r1
has_location f1 k1
has_location table-1 r2
has_position mb1 table-1
nav b1 r1
nav k1 r1
nav r2 r1
nav r3 r1
```

Figure 6: Factual information.

Norm violation detection. The inclusion of the above factual information into the Pellet knowledge representation system causes a number of inconsistencies, which are detected by Pellet as follows:

- Violation #1: a towel (`t1`) is located in a kitchen (`k1`). This violation results in (`k1`) being inferred to be an instance of two mutually exclusive concepts, and the reasoner catches this inconsistency as seen in Sec. 4.3 above.
- Violation #2: a towel (`t2`) is located in a livingroom (`r1`). This is a variation of violation #1.
- Violation #3: a shoe (`o1`) is located in a kitchen (`k1`). Like in the previous cases, Pellet deduces that (`k1`) is an instance of the two mutually exclusive concepts `Bedroom` (since a shoe is located in it) and `Kitchen` (since it has so been asserted). This inconsistency is catched by the reasoner as:

  ```
  [disjointWith(Kitchen, Bedroom),
   subClassOf(Shoe, all(has-location, Bedroom)),
   prop(has-location, o1, k1),
   type(o1, Shoe),
   type(k1, Kitchen)]
  ```

- Violation #4: a bottle of milk (`mb1`) is on a table (`table-1`). Again, this results in an object (`table-1`) being inferred to be an instance of two mutually exclusive concepts: `Frigde` and `Table`. This inconsistency is catched by the reasoner as:

  ```
  [disjointWith(Fridge, Table),
   subClassOf(Milk-Bottle, all(has-position, Fridge)),
   prop(has-position, mb1, table-1),
   type(mb1, Milk-Bottle),
   type(table-1, Table)]
  ```

Goal generation and planning. As noted above, these inconsistencies cannot all be detected at the same time: once an inconsistency is detected, e.g., the one produced by *violation #1*, the whole knowledge base becomes inconsistent and no further inferences can be done. We therefore analyze inconsistencies one by one by repeatedly: (1) detecting the first inconsistency reported by the reasoner, and (2) removing the cause of that inconsistency from the knowledge base. These steps are repeated until the knowledge base become consistent. For each detected inconsistency, the generation of the appropriated goal is done as discussed in Sec. 4.3 above: for violations #1, #2 and #3, the selected goal is of type `(bring ?x ?y)`; analogously, for violation #4, the selected goal is of type `(put-on ?x ?y)`.

Priorities. In order to prioritize the planning and execution of multiple violations, we compute a priority value in the range $[0, 1]$ for each generated goal, hence for each violated normative constraint. The computation is based on the fixed priority values associated to the normative concepts and relations, set by the domain engineer. In our example, we have used the following values:

Concept	Priority
Towel	0.85
Shoe	0.5
MilkBox	1.0

Relation	Priority
has_location	0.9
has_position	1.0

These values are obviously naive, and ideally they should be learned by the robot. Nonetheless, they serve the purpose of illustrating our approach. With these values, the priority of each generated goal is:

Generated goal	Priority
(bring t1 b1)	$0.9 * 0.85 = 0.765$
(bring t2 b1)	$0.9 * 0.85 = 0.765$
(bring o1 r2)	$0.9 * 0.5 = 0.45$
(put-on mb1 f1)	$1.0 * 1.0 = 1.0$

From this set of goals and priorities, we generate and execute the corresponding plans in the order given by the priority values. In our example, the robot will first put the milk bottle inside the fridge, then bring the towels to the bathroom, and finally it will bring the shoe to the bedroom. The complete log for this example is shown in Appendix A.

6. An Integrated Experiment

In order to test the suitability of our approach in a real robotic application, we have run a few proof-of-concept experiments in which our goal generation system has been integrated in an existing robotic application. In this section, we describe one such experiment. The experiment only deals with a single norm violation, since the case of multiple violations does not add any complexity from the point of view of robotic execution.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

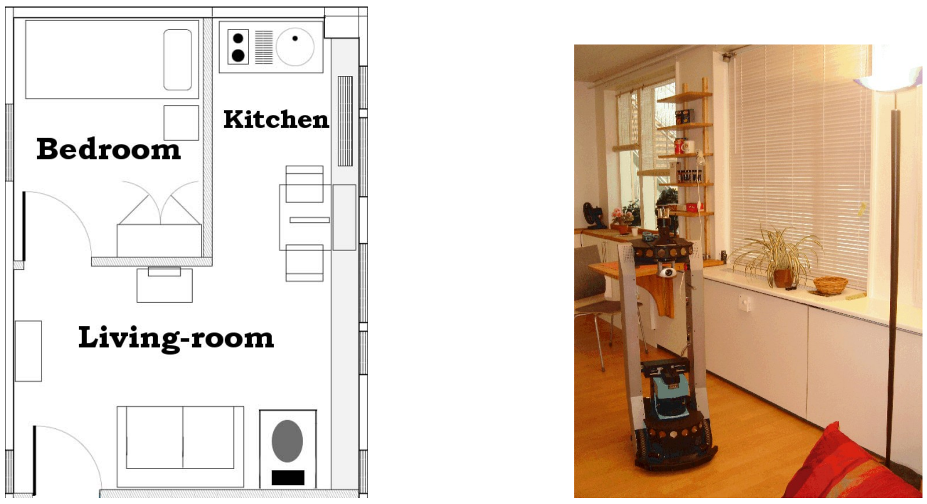


Figure 7: The test environment. Left: layout. Right: the robot Astrid.

6.1. Physical setup

We have used a test-bed facility, called the PEIS-Home [46]. The environment looks like a bachelor apartment of about $25\,m^2$ and consists of a living-room, a bedroom and a small kitchen — see Fig. 7. The PEIS-Home is equipped with a communication infrastructure, and with a number of sensing and actuating devices, including a few mobile robots. Relevant to the experiment reported here are:

- a refrigerator equipped with a computer, some gas sensors, a motorized door, and an RFID tag reader;
- an RFID tag reader mounted under the kitchen table;
- a set of RFID tagged objects, including a milk box;
- a set of webcams mounted on the ceiling; and
- Astrid, a PeopleBot mobile robot equipped with a laser scanner, a PTZ camera, and a simple gripper.

A few provisions have been introduced to simplify perception and execution. In particular, since Astrid does not have a manipulator able to pick-up an object and place it somewhere else, these operations have been performed with the assistance of a human who puts the object in and out from the Astrid's gripper. Also, we have exploited the RFID-based object tracking system present in the PEIS-Home to simplify object recognition. These simplifications are acceptable here, since the purpose of our experiment is not to validate the perception and execution system, but to illustrate the integration of our goal generation approach into a full robotic application.

6.2. Software setup

The software system used in this experiment relies on the software already in use in the PEIS-Home, and it is schematically shown in Fig. 8. The block named "PEIS Ecology" contains all the robotic components and devices distributed in the PEIS-Home.

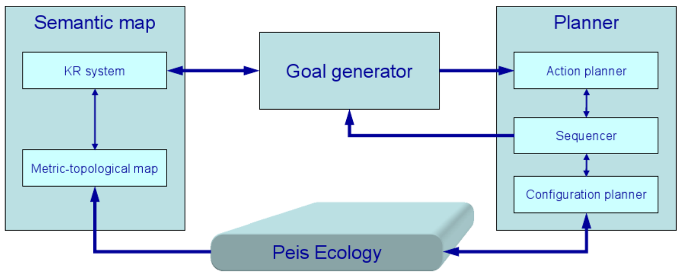


Figure 8: Sketch of the software architecture used in our experiments. Only the modules and connections relevant to goal generation are shown.

These are integrated through a specific middleware, called the PEIS-Middleware, that allows to dynamically activate and connect them in different ways in order to perform different tasks [47]. A set of activations and connections is called a *configuration* of the PEIS Ecology. For instance, the configuration in which the ceiling cameras are connected via an object recognition to the navigation controller onboard Astrid can be used to let the robot reach a given object.

The knowledge representation system and the planning system are similar to the ones used in the previous sections. The planning system, however, also includes a configuration planner [48], that generates the configuration needed to cooperatively perform each action in the PEIS-Home. When the current plan is completed, the goal generation system is re-activated.

6.3. Execution

Before the execution started, the semantic map contained a metric-topological map of the PEIS-Home, and the considered semantic knowledge. In particular, this includes the normative constraint that milk should be placed in a fridge, as shown in Fig. 4 above.

An RFID tag has been attached to a milk box, containing an encoding of the following information:

```
id: mb-22
type: MilkBox
color: white-green
size: 1-liter
```

At start, the milk is put on the kitchen table, called `table-1` in the map. The RFID tag reader under the table detects the new tag, and reports the information that `mb-22` is a `MilkBox` and it is at `table-1` — see Fig. 9. This information is entered into the semantic map by asserting the following facts into the KR system:

```
 (MilkBox mb-22)
 (place mb-22 table-1)
```

This information results in an inconsistency, since `table-1` is deduced to be an instance of both `Frigde` and `Table`. The goal generation algorithm generates the following goal, as shown in the previous section:

```
  (put-on mb-22 fridge-1)
```

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

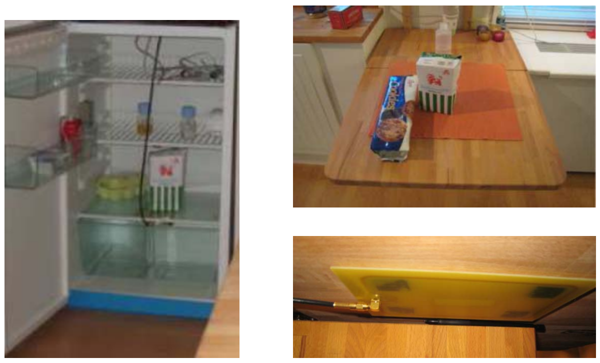


Figure 9: Left: the fridge used in our experiment, equipped with a door opening mechanism and an RFID tag reader. Right: the RFID tagged milk box on the kitchen table, equipped with a second RFID tag reader.

The planner uses the factual knowledge in the semantic map, together with its domain knowledge about the available actions and methods, to generate the following action plan (simplified):

```
(!approach astrid table-1)
(!pickup astrid mb-22 table-1)
(!open fridge-1)
(!approach astrid fridge-1)
(!drop astrid mb-22 fridge-1)
(!close fridge-1)
```

The sequencer passes each action in turn to the configuration planner, which connects and activates the devices in the PEIS Ecology needed to execute it. For example, the first two actions only require devices which are on-board Astrid, while the third action requires the activation of the fridge door device. (The details of this "ecological" execution are not relevant here: see [48] for a comprehensive account.) As mentioned above, the `pickup` and `drop` actions were performed with the help of a human.

After the milk is removed from the table, the RFID tag reader under the table detects its absence and it signals it to the semantic map. When the milk is placed into the fridge, it is detected by the reader in the fridge. Corresponding to these two events, the fact `(place mb-22 table-1)` is removed from the KR system, and the new fact `(place mb-22 fridge-1)` is asserted. After execution is completed, the sequencer re-activates the goal generator. Since the place of `mb-22` is now an instance of a fridge, no incoherence is detected and no new goal is generated.

7. Discussion and Conclusions

One of the most promising uses of semantic knowledge in a robotic system is to resolve situations of conflict or ambiguity by reasoning about the cause of the problem and its possible solutions. This paper has explored an often neglected aspect of this use: recognizing and correcting situations in the world that do not comply with the given semantic model, by generating appropriate goals for the robot. A distinctive feature of our approach is that the normative model is provided in a declarative way, rather than by exhaustive violation-action rules. Experiments carried out on an integrated robotic system demonstrate the conceptual viability of this approach.

The work reported here is a first step in an interesting direction, and many extensions can and should be considered. In particular, in our work we assume that the robot should *always* enforce consistency with the semantic knowledge. However, there are cases where norm violation might be allowed. Going back to the milk example, it would be reasonable to allow that the milk bottle stays out of the fridge for some amount of time while the user is having breakfast. We believe that our scheme for automatic goal generation can be extended to also cope with this and other issues by introducing the notion of *context*, that is, different semantic representations of the environment that can be adopted by the robot according to external events.

References

[1] J. Hertzberg, A. Saffiotti (Eds.), Special issue on semantic knowledge in robotics, Robotics and Autonomous Systems 56 (11).

[2] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robotics and Autonomous Systems 56 (11) (2008) 915–926.

[3] D. Meger, P. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. Little, D. Lowe, Curious george: An attentive semantic robot, Robotics and Autonomous Systems 56 (6) (2008) 503–511.

[4] O. Mozos, P. Jensfelt, H. Zender, M. Kruijff, W. Burgard, From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots, in: ICRA Workshop: Semantic Information in Robotics, 2007.

[5] C. Galindo, J. Fernandez-Madrigal, J. Gonzalez, A. Saffiotti, Robot task planning using semantic maps, Robotics and Autonomous Systems 56 (11) (2008) 955–966.

[6] A. Ranganathan, F. Dellaert, Semantic modeling of places using objects, in: Proc of Robotics: Science and Systems, 2007.

[7] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernandez-Madrigal, J. Gonzalez, Multi-hierarchical semantic maps for mobile robotics, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS, Edmonton, Alberta (Canada), 2005, pp. 3492–3497.

[8] A. Bouguerra, L. Karlsson, A. Saffiotti, Monitoring the execution of robot plans using semantic knowledge, Robotics and Autonomous Systems 56 (11) (2008) 942–954.

[9] F. Baader, D. Calvanese, D. McGuinness, D. Nardi (Eds.), The Description Logic Handbook, Cambridge University Press, 2007.

[10] C. Galindo, J. González, J. Fernández-Madrigal, A. Saffiotti, Robots that change their world: Inferring goals from semantic knowledge, in: Proc of the European Conf on Mobile Robotics (ECMR), 2011, pp. 1–6.

[11] D. Holz, D. Munoz, A. Nüchter, R. Bogdan-Rusu (Eds.), Workshop on Semantic Perception, Mapping and Exploration, 2011, Int Conf on Robotics and Automation.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

[12] M. Beetz, R. Alami, J. Hertzberg, A. Saffiotti, M. Tenorth (Eds.), Workshop on Knowledge Representation for Autonomous Robots, 2011, Int Conf on Robotics and Automation.

[13] S. Vasudevan, S. Gächter, V. Nguyen, R. Siegwart, Cognitive maps for mobile robots-an object based approach, Robotics and Autonomous Systems 55 (2007) 359–371.

[14] A. Swadzba, S. Wachsmuth, C. Vorwerg, G. Rickheit, A computational model for the alignment of hierarchical scene representations in human-robot interaction, in: Proc of the Int Joint Conf on Artificial Intelligence (IJCAI), 2009, pp. 1857–1863.

[15] O. Mozos, C. Stachniss, W. Burgard, Supervised learning of places from range data using adaboost, in: Proc of the Int Conf on Robotics and Automation (ICRA), 2005, pp. 1730–1735.

[16] A. Pronobis, O. Martinez-Mozos, B. Caputo, P. Jensfelt, Multimodal semantic place classification, Int Journal of Robotics Research 29 (2-3) (2010) 298–320.

[17] J. Civera, D. Gálvez-Lóopez, L. Riazuelo, J. Tardós, J. Montiel, Towards semantic SLAM using a monocular camera, in: Proc of the Int Conf on Intelligent Robots and Systems (IROS), 2011.

[18] F. Dayoub, T. Duckett, G. Cielniak, Toward an object-based semantic memory for long-term operation of mobile service robots, in: IROS Workshop on Semantic Mapping and Autonomous Knowledge Acquisition, 2010.

[19] H. Zender, O. Martínez-Mozos, P. Jensfelt, G. Kruijff, W. Burgard, Conceptual spatial representations for indoor mobile robots, Robotics and Autonomous Systems 56 (6) (2008) 493–502.

[20] R. Rusu, Z. Marton, N. Blodow, M. Dolha, M. Beetz, Towards 3D point cloud based object maps for household environments, Robotics and Autonomous Systems 56 (11) (2008) 927–941.

[21] J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G. Kruijff, P. Lison, A. Pronobis, K. Sjöö, A. Vrecko, H. Zender, M. Zillich, D. Skocaj, Self-understanding & self-extension: a systems and representational approach, IEEE Trans on Autonomous Mental Development 2 (4) (2010) 282–303.

[22] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Haussermann, J. Montiel, A. Perzylo, B. Schiele, O. Zweigle, R. van de Molengraft, Roboearth – a world wide web for robots, IEEE Robotics and Automation Magazine 18 (2) (2011) 69–82.

[23] M. Wooldridge, N. Jennings, Intelligent Agents, Springer-Verlag, 1995.

[24] C. Martin, K. Barber, Agent autonomy: Specification, measurement, and dynamic adjustment, in: Proc of the Autonomy Control Software Workshop at Autonomous Agents, 1999, pp. 8–15.

[25] R. Conte, C. Castelfranchi, Cognitive and social action, University College of London Press, London, 1995.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

[26] M. Dastani, L. W. N. van der Torre, What is a normative goal?: Towards goal-based normative agent architectures, in: Regulated Agent-Based Social Systems, First International Workshop (RASTA), 2002, pp. 210–227.

[27] G. Boella, R. Damiano, An architecture for normative reactive agents, in: Proc of the 5th Pacific Rim International Workshop on Multi Agents, 2002, pp. 1–17.

[28] M. Wooldrige, An Introduction to MultiAgent Systems, 2nd Edition, Wiley, New York, 2009.

[29] R. Arkin, M. Fujita, T. Takagi, R. Hasegawa, An ethological and emotional basis for human-robot interaction, Robotics and Autonomous Systems 42 (3-4) (2003) 92–112.

[30] A. M. Coddington, M. Fox, J. Gough, D. Long, I. Serina, MADbot: A motivated and goal directed robot, in: Proc of the 12th Nat Conf on Artificial Intelligence, 2005, pp. 1680–1681.

[31] T. J. Norman, D. Long, Alarms: An implementation of motivated agency, in: Intelligent Agents II, Agent Theories, Architectures, and Languages, IJCAI '95, Workshop (ATAL), 1995, pp. 219–234.

[32] C. Baral, T. Eiter, M. Bjäreland, M. Nakamura, Maintenance goals of agents in a dynamic environment: Formulation and policy construction, Artif. Intell. 172 (12-13) (2008) 1429–1469.

[33] K. V. Hindriks, M. B. van Riemsdijk, Satisfying maintenance goals, in: Declarative Agent Languages and Technologies V, 5th International Workshop (DALT), 2007, pp. 86–103.

[34] M. Dean, G. Schreiber, OWL web ontology language reference, http://www.w3.org/TR/2004/REC-owl-ref-20040210/ (2004).

[35] B. Kuiper, Modeling Spatial Knowledge, The University of Chicago Press, 1990, Ch. Advances in Spatial Reasoning, Vol. 2, pp. 171–198.

[36] P. Buschka, A. Saffiotti, Some notes on the use of hybrid maps for mobile robots, in: Proc of the 8th Int Conf on Intelligent Autonomous Systems (IAS), Amsterdam, NL, 2004, pp. 547–556.

[37] C. Galindo, J. Fernandez-Madrigal, J. Gonzalez, Multiple Abstraction Hierarchies for Mobile Robot Operation in Large Environments, Studies in Computational Intelligence, Vol. 68. Springer, 2007.

[38] A. Bouguerra, L. Karlsson, A. Saffiotti, Semantic knowledge-based execution monitoring for mobile robots, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, Rome, Italy, 2007, pp. 3693–3698.

[39] E. Sirin, B. Parsia, B. Cuenca-Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semantics: Science, Services and Agents on the World Wide Web 5 (2) (2007) 51–53.

[40] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The description logic handbook: theory, implementation, and applications, Cambridge University Press, New York, NY, USA, 2003.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

[41] Protegé, Home page, `http://protege.stanford.edu/`.

[42] O. Ilghami, D. S. Nau, A general approach to synthesize problem-specific planners, Tech. Rep. CS-TR-4597 UMIACS-TR-2004-40, University of Maryland (2003).

[43] D. Nau, M. Ghallab, P. Traverso, Automated Planning: Theory & Practice, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[44] M. van den Briel, R. Sanchez, S. Kambhampati, Over-subscription in planning: A partial satisfaction problem, in: Proc of the ICAPS Workshop on Integrating Planning into Scheduling, 2004, pp. 91–98.

[45] M. Cirillo, L. Karlsson, A. Saffiotti, A human-aware robot task planner, in: Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS), 2009, pp. 58–65.

[46] A. Saffiotti, M. Broxvall, M. Gritti, K. LeBlanc, R. Lundh, J. Rashid, B. Seo, Y. Cho, The PEIS-ecology project: vision and results, in: Proc of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS), 2008, pp. 2329–2335.

[47] M. Bordignon, J. Rashid, M. Broxvall, A. Saffiotti, Seamless integration of robots and tiny embedded devices in a PEIS-ecology, in: Proc of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS), 2007, pp. 3101–3106.

[48] R. Lundh, L. Karlsson, A. Saffiotti, Autonomous functional configuration of a network robot system, Robotics and Autonomous Systems 56 (10) (2008) 819–830.

Draft Version
Final version published in Robotics and Autonomous Systems, 2013

Appendix A. Console Output for Multiple-Violations Example

```
Loading ontology file:owl/goalgeneration.owl
Loading problem from file src\domain\facts.txt
Goal generator subsystem starts...
Detected violation: Milk_Bottle has_position Fridge
Goal proposed: put-on mb1 f1
Detected violation: Towel has_location Bathroom
Goal proposed: bring t1 b1
Detected violation: Towel has_location Bathroom
Goal proposed: bring t2 b1
Detected violation: Shoe has_location Bedroom
Goal proposed: bring o1 r2
-------------------------
Preparing planning domain
-------------------------
Solving violation with priority: 1.0

put-on mb1 f1
Plan cost: 6.0

(!approach table-1)
(!pickup mb1 table-1)
(!move r2 r1)
(!move r1 k1)
(!approach f1)
(!drop mb1 f1)
--------------------


bring t2 b1
Solving violation with priority: 0.765
Plan cost: 4.0

(!move k1 r1)
(!pickup t2 r1)
(!move r1 b1)
(!drop t2 b1)
--------------------

Solving violation with priority: 0.765

bring t1 b1
Plan cost: 6.0

(!move b1 r1)
(!move r1 k1)
(!pickup t1 k1)
(!move k1 r1)
(!move r1 b1)
(!drop t1 b1)
--------------------

Solving violation with priority: 0.45

bring o1 r2
Plan cost: 6.0

(!move b1 r1)
(!move r1 k1)
(!pickup o1 k1)
(!move k1 r1)
(!move r1 r2)
```

```
(!drop o1 r2)
--------------------

Consistent world is reached...
```

Appendix B. Planning Domain

```
(defdomain servant-robot (

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Operators
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:operator (!move ?x ?y)
  ((has_location robot-1 ?x) (not (visited ?y)))
  ((has_location robot-1 ?x))
  ((has_location robot-1 ?y)(visited ?y))
)

(:operator (!unvisit)
  ()
  ()
  ((forall (?x ?y) ((nav ?x ?y) (not (has_location robot-1 ?y)))
           ((visited ?y))))
)

(:operator (!approach ?x)
  ()
  ()
  ()
)

(:operator (!pickup ?object ?location)
  ()
  ((has_location ?object ?location)
        (forall (?x ?y) ((nav ?x ?y) (not (has_location robot-1 ?y)))
                ((visited ?y))))
  ((have robot-1 ?object))
)

(:operator (!drop ?object ?location)
  ()
  ((have robot-1 ?object)
          (forall (?x ?y) ((nav ?x ?y) (not (has_location robot-1 ?y)))
                  ((visited ?y))))
  ((has_location ?object ?location))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Methods
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:method (go ?x ?y)
   (has_location robot ?y) ()
   (and (has_location robot-1 ?x) (nav ?x ?y)) ((!move ?x ?y))
   (and (has_location robot-1 ?x) (not (nav ?x ?y)) (nav ?x ?z)
                  (not (visited ?z))) ((!move ?x ?z)(go ?z ?y))
)

(:method (bring ?x ?y)
   ((has_location ?x ?z)) ((take ?x ?z) (go ?z ?y) (!drop ?x ?y))
)

(:method (take ?x ?y)
   ((has_location ?x ?y) (has_location robot-1 ?z))
                        ((go ?z ?y) (!pickup ?x ?y))
)

(:method (take-from ?x ?y)
   ((has_position ?x ?y) (has_location ?y ?z) (has_location robot-1 ?z))
                            ((!approach ?y) (!pickup ?x ?y))
   ((has_position ?x ?y) (has_location ?y ?z) (has_location robot-1 ?w))
                             ((go ?w ?z) (!approach ?y) (!pickup ?x ?y))
)

(:method (put-on ?x ?y)
    ((has_position ?x ?q)(has_location ?y ?z)(has_location ?q ?w ))
        ((take-from ?x ?q) (go ?w ?z) (!approach ?y) (!drop ?x ?y))
)

))
```