Draft Version
Published in 10th International IFAC Symposium on Robot Control, Dubrovnik, 2012

Semantic Norms for Mobile Robots: When the end does not justify the means

C. Galindo * **A. Saffiotti** **

* *System Engineering and Automation Dept*
University of Málaga, Spain (e-mail: cipriano@ctima.uma.es)
** *AASS Cognitive Robotic Systems Lab*
Örebro University, Sweden (e-mail: asaffio@aass.oru.se)

Abstract: This paper deals with the use of semantic knowledge to improve the intelligence and autonomous behavior of a mobile robot. A robot can exploit the semantics of its environment to infer new, implicit information. Another interesting possibility is to use semantics for detecting deviations between the real world and what is supposed to be "normal". For instance, normative semantic knowledge may state that towels should stay in the bathroom. If a robot detects a towel in the kitchen, it can react and decide to solve this inconsistency by bringing it to the bathroom. However not all ways to solve an inconsistency are acceptable: for instance, if the robot put the towel temporarily on a dirty sink in order to re-grasp it with the other arm, it would violate another norm – namely, that towels should always stay on a clean surface. In this work we present an algorithm that detects and recovers from norm violations, according to a semantic representation of norms, and ensures the normative acceptability of the robot actions throughout execution.

Keywords: Semantic maps, Mobile robotics, Goal autonomy, Normative constraints, Knowledge representation, Description logic, Ontology, Planning.

1. INTRODUCTION

The use of semantic knowledge has emerged as a required ingredient for successful mobile robotic applications. Concretely, service robots are being increasingly endowed with the ability to represent and use semantic knowledge about the environment where they operate – see, e.g., J. Hertzberg, A. Saffiotti (Eds.) (2008).

Semantic knowledge encodes general and commonsense information about the elements of the world and their relations. Semantic knowledge can tell, for example, that a bathroom is a type of room which typically contains a bathtub and a basin; that a towel is an element related to the personal hygiene; and that this type of elements are located in a bathroom. A mobile robot able to manage such information will show improved autonomy and more intelligent behavior. For example, the robot could classify a room as being a bathroom if it perceives a bathtub in it; it could infer the most likely location to find a towel; and it could detect an abnormal situation, like observing a bathtub in what is believed to be a living room, and react accordingly. Several works have been reported in the literature which have addressed these issues from different perspectives, e.g., by Nüchter et al. (2005); Meger et al. (2007); Mozos et al. (2007); Ranganathan and Dellaert (2007); Zender et al. (2008); Galindo et al. (2008, 2011); Pronobis (2011).

⋆ Corresponding author. System Engineering and Automation Dpt. University of Málaga, Campus de Teatinos. E-29071 Málaga, Spain. Email: cipriano@ctima.uma.es

A somehow less studied facet of semantic knowledge is its use as a source of normative information on how the world should be. In this paper, we investigate this facet building upon our previous work (Galindo et al., 2011), which addresses goal autonomy through the proactive generation of goals based on the robot's internal semantic model. In that work, we considered a robot with the innate objective of keeping its environment in good order with respect to a given set of norms. Norms were encoded in a declarative way in a semantic representation, enabling the robot's reasoning systems to automatically detect violations of a particular norm, by translating violations into logical incoherences between the sensed facts and the model. For example, if the robot finds a towel inside the kitchen, it infers a violation of the commonsense piece of knowledge that states that towels are elements related to the personal hygiene and therefore they are located in a bathroom. The output of the system consisted of the generation of a robot goal that, when planned and executed, put the world in a state consistent with the normative knowledge. In the previous example, the robot would generate the goal to bring the towel from the kitchen to the bathroom.

The above work provides, to the best of our knowledge, the first example of a robot which uses semantic, ontological knowledge to both describe and maintain the nominal state of the world. That work has, however, an important limitation: when generating a plan to recover from the violation of a norm, the robot does not consider that the execution of this plan may in turn result in the violation of other norms. Simply put, the above approach adheres to Macchiavelli's principle *the end justifies the means*. To

see why this can be a problem, consider the above norm that towels should stay in the bathrooms, together with the second norm that towels (being items of personal hygiene) should be put on clean surfaces. Imagine that the robot, who has inferred the goal to bring the towel to the bathroom, generates a corresponding plan to grasp the towel, move to the bathroom, and place to towel. Imagine also that, as part of this plan, the robot places the towel temporarily on the sink, turns, and re-grasps it with the other arm – this may be needed because of the geometric constraints of the environment. Putting the towel on the sink, however, violates the norm that states that towels should stay on clean surfaces. Thus, while the robot tries to correct a norm violation, it is violating another one: in some cases, *the end does not justify the means*.

In this paper we present a solution to the above problem. We propose an algorithm that checks the normative admissibility of the generated plans, and interacts with the planner in order to discard inadmissible plans and generate new ones. We also introduce the notion of *transient* norms, i.e., norms which can be temporarily violated during the execution of the plan, as opposed to *permanent* norms, i.e., norms that must be obeyed at all times. In the above example, the location of the towel is a transient norm, since it is acceptable that the towel is temporarily in the corridor during the motion; while the clean support requirement is a permanent norm, since it is not acceptable to violate it even in a transient way.

This manuscript is structured as follows. The next two sections introduce the main ingredients for our approach: our semantic maps, and our approach to detect and recover from norm violations. Section 4 presents our new algorithm to generate recovery plans that do not incur in norm violations. An implementation and illustrative example of this algorithm are described in section 5. Finally some conclusions and future work are outlined.

2. A SEMANTIC MAP FOR MOBILE ROBOT OPERATION

A mobile robot requires a proper representation of its workspace, i.e., a map, in order to perform with a certain autonomy. The characteristics of the data stored in such a map will determine the robot abilities. Geometrical and topological maps enable a mobile robot to perform localization and navigation tasks; symbolic maps are needed for task planning, i.e. search for a sequence of robot actions to achieve a given goal. In order to improve the robot skills with *semantic information*, more sophisticated maps are required. The map we considered in this work is derived from the world presented by Galindo et al. (2008). It 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 geometrical, topological and symbolic *information about the current state* of the environment and of the objects inside it. On the other hand, the T-Box contains *general semantic knowledge* about the 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 and that a stove is a type of appliance. By combining

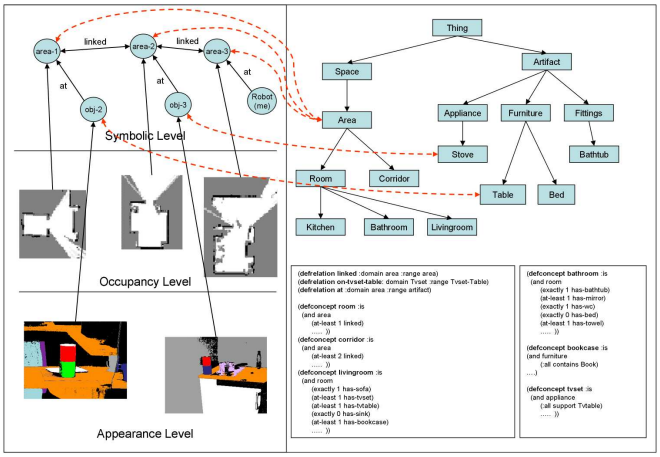

Fig. 1. An example of semantic map for a home-like environment. S-Box is on the left and T-Box on the right. See explanation in the text.

the two sides, the semantic map can infer, for instance, that `Area-2` is a kitchen, since it contains a stove.

This structure is reminiscent of hybrid knowledge representation (KR) systems (Brachman and Levesque, 2004; Baader et al., 2007), which are dominant in the KR community. 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. See Galindo et al. (2008) for more detail.

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. For the S-Box, the use of a hierarchical spatial representation is a convenient and common choice in the robotic literature for dealing efficiently with large-scale environments (Kuiper, 1990; Galindo et al., 2007). 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.

The semantic map enables a mobile robot to reason about its environment through the T-box, linking conceptual, inferred knowledge, to perceived world instances it the S-Box, which hold the needed sensorial information for planning and execution. This map can also be used to endow the robot with the ability to detect incoherences between both sides of the map and act accordingly. The next two sections will show how to use this feature to (i) detect and resolve norm violations, and (ii) anticipate the effects of actions to avoid new violations.

3. DOMAIN CONSISTENCY

An interesting use of our semantic maps is to detect contradictions between the general knowledge encoded in the map, and the specific situations observed by the robot. This use has been previously explored, e.g., by Bouguerra et al. (2008); Galindo et al. (2008). The generic knowledge represented in the semantic map can also be used to encode

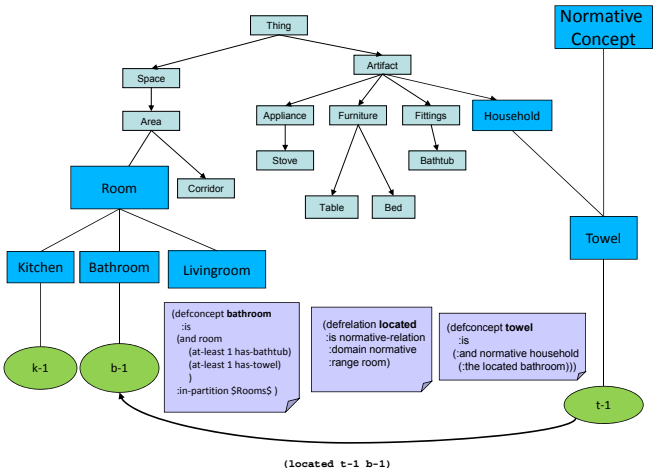

Fig. 2. Example of norm violation. Towels are supposed to be at bathrooms. Asserting a different location for a given towel makes the knowledge base inconsistent.

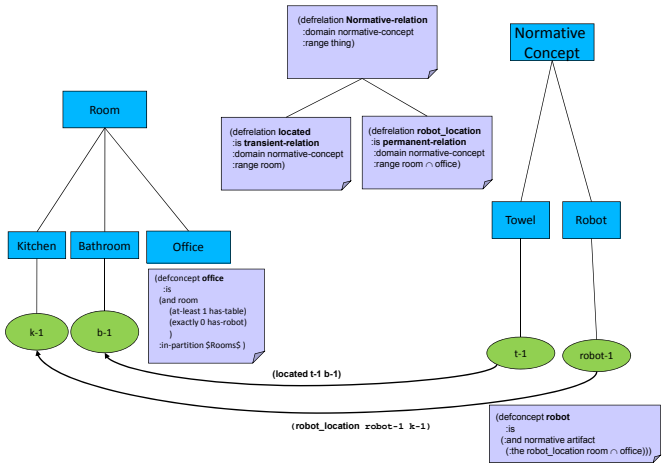

Fig. 3. Normative relation taxonomy. Transient and permanent relations are normative relations involving normative concepts.

normative knowledge, which describes the proper and valid relations between objects in the considered domain. In this case, a contradiction in the map reflects an observed violation of a norm, that is, a failure to comply with an established regulation within the given domain.

Galindo et al. (2011) proposes an approach to represent and resolve norm violations based on the above idea. In order to make this paper self-contained, we provide here a basic summary of this approach.

We represent normative knowledge by introducing two special entities in the ontology defined in the T-Box: a concept called `Normative-concept`, that subsumes all the concepts that are involved in a norm; and a relation called `normative-relation`, that subsumes all the norms that have to be fulfilled.

We can use these entities to encode, for instance, the item of normative knowledge that *towels have to be located in a bathroom* as follows: we define the concept `Towel` as being of type `Normative-concept`, and we state that towels are related to the concept `Bathroom` through the relation `located`, which is of type `normative-relation`.

More specifically, norms are represented in our system as relations whose domains are normative concepts and their ranges belong to a mutually disjoint set of concepts. For the towel example, the relation `located` has the normative concept `Towel` as domain and the set of `rooms` as range, that is the set of possible locations for objects in the considered scenario (see figure 2). Note that `rooms` is defined to be a disjoint set, that is, its sub-concepts `Kitchen`, `Bathroom` and `Livingroom` are disjoint.

The way in which normative information is encoded in the T-Box permits its automatic detection by most of knowledge representation and reasoning systems, including the *Pellet* reasoner used in the experiments reported below. The violation of a norm is automatically identified when a given instance that belongs to a normative concept is related, through a normative relation, to an instance of a not-allowed concept. Given that the range of the normative relations is defined as a disjoint set, the inference system concludes that there is an *incoherence* since an instance belongs to two disjoint concepts.

For example, suppose that an instance of a towel, `t-1`, is perceived in a room, `k-1`, which has been previously asserted to be an instance of the concept `Kitchen`. Then, the piece of knowledge `(located t-1 k-1)` is asserted into semantic map. In this situation, the T-Box inference system identifies `k-1` as an instance of a `Bathroom`, since this is the range of the normative relation `located` when applied to towels. As a result, the instance `k-1` is inferred to belong to both the `Kitchen` and `Bathroom` concepts. These two concepts, however, have been defined as being mutually disjoint, which causes the system to be in a state of logically inconsistence: this incoherence makes the violation of the norm explicitly visible in the system.

In order to recover from the incoherence, and therefore from the detected norm violation, the system by Galindo et al. (2011) generates the goal to satisfy the violated constraint, that is, to make the location of towel `b-1` be an instance of the correct range, `Bathroom`, instead of `Kitchen`. If the robot knows that, let say, `b-1` is a bathroom, then the system generates the goal to *bring t-1 from k-1 to b-1*. This goal is then passed to a planner, which generates a suitable plan using the information about the current state contained in the S-Box. When executed, this plan will make the goal hold, thus solving the detected violation.

4. NORMS VIOLATION DURING EXECUTION

The approach presented in the previous section ensures that the norm violation that started the process is solved at the end of the plan execution. However, it does not ensure that the plan execution itself does not produce new norm violations, either at the goal state or at some intermediate state. In this section we define an algorithm to detect, at the planning phase, plans whose executions will result in norm violations, and to reject those plans.

4.1 Permanent vs. Transient Norms

Recall the towel example given in the Introduction. When executing the plan to bring the towel back to the bathroom, the robot may cause two types of norm violations: (i) the towel will temporarily be in the places traversed by the robot (e.g., the corridor), which contradicts the norm that towels should be in bathrooms; (ii) the towel will temporarily be on the the sink, which contradicts the norm that towels should be placed on clean surfaces. Both these violations are temporary: at the end of the plan the towel will be on its hanger in the bathroom, and no norm is violated in the final state. However, while violation (i) is intuitively acceptable (and inevitable), violation (ii) is not since it concerns a norm which should never be violated, even for a short moment.

In order to capture the distinction between the above two cases, we distinguish two types of norms: *permanent* and *transient* norms. We define transient norms, or T_{norms}, as a type of norms that can be violated during the execution of a robot plan, but should be met at the final state, i.e., when the plan terminates. Trying to solve the violation of a transient norm during plan execution would be pointless, since the norm will be met at the final state. On the other hand, we define permanent norms, or P_{norms}, as norms that must be abided to at any time – see figure 3. Violations of permanent norms should be avoided rather than repaired. Correspondingly, any plan whose execution will predictably incur in the (temporary) violation of a permanent norm must be rejected.

In the towel example above, the requirement that the towel is in a bathroom should be treated as a transient norm, while the requirement that the towel is on a clean surface should be treated as a permanent one. As another example, consider a permanent norm that forbids the robot to enter into the office for any reason: then, no plan should include a navigation action through the office, and alternative plans should be sought.

4.2 Solving Norm Violations

We now show how to detect, at planning time, plans that will result in norm violations. The general idea is to analyze the plans proposed by the planner in order to check the effects of individual actions, and to reject those plans, or individual actions, which result in a violation. First we define our planning problem, and then we present our algorithm for anticipating and resolving norm violations.

Let $\wp$ be a planning problem, defined as $\wp = (s_i, s_g, O)$ where s_i is the initial state, s_g is the goal state and O is a set of operators which define the abstract actions available to the robot. The aim of the planning system is to generate a plan, i.e., a sequence of ordered instantiated operators, or actions, $(o_1, o_2, \ldots, o_n)$, that transforms the initial state, s_i, in a final state s_f such that $s_g \subseteq s_f$.

Each operator $o_i \in O$ is a 3-tuple $o_i = (pre, add, del)$ where *pre* is the set of preconditions that has to hold on a given state s in order to apply o_i, and *add* and *del* are the postconditions for the operator, that is, the sets of statements added or removed from the state s after the application of o_i. (We direct the reader unfamiliar with planning to Ghallab et al. (2004).)

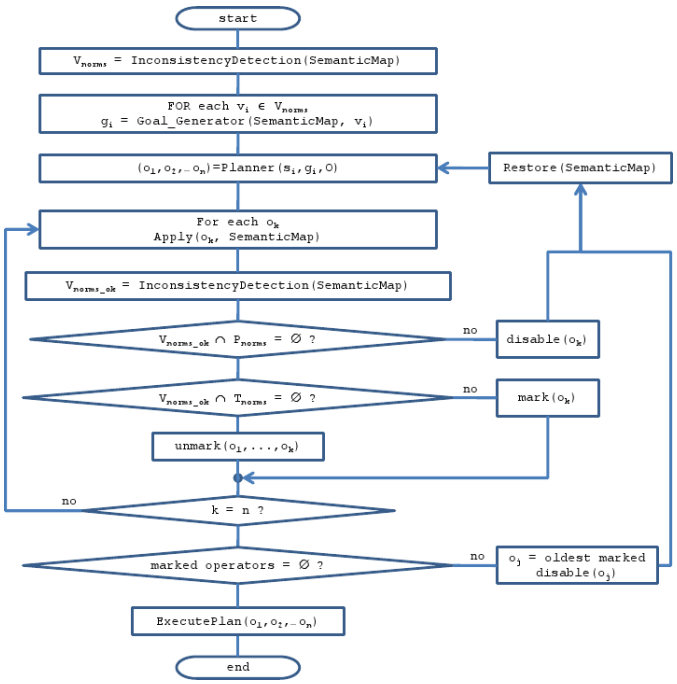

Fig. 4. Algorithm for the detection of norms violations. The effect of each action is sequentially checked. Only violation of transient norms are allowed at the intermediate steps.

Our algorithm for maintaining norm consistency is shown in figure 4. This extends the algorithm proposed by Galindo et al. (2011), and accordingly the first three steps are the same: the violated norms are detected, the corresponding goals are generated, and a planner is invoked to produce suitable plans.

The new part starts at step 4. Given a plan $(o_1, o_2, \ldots, o_n)$, we sequentially analyze each action $o_k, k = 1, \ldots, n$, and simulate the effect of executing this action. Simulation is done by adding and removing to the semantic map the operator's *add* and *del* postconditions, respectively. Then, the inconsistency detection mechanism is executed to check the integrity of the knowledge base. If a permanent norm is violated, then the plan is rejected and the planner is asked to generate a new plan. Before the planner is invoked, the semantic map is restored to its state before the actions were simulated; also, action o_k is disabled so the planner will not include it in the new plan.

If a transient norm is violated after simulating the execution of o_k, the algorithm continues with the next action in the plan. However, o_k is marked as a possible culprit for a violation in the final state: if any action is so marked after the algorithm has analyzed all the actions, this means that a transient norm is violated in the final state, which is not acceptable. In this case, the planner is given the oldest culprit, and is requested to find an alternative plan that does not involve that action.

If no norms are violated after simulating the execution of o_k, then the list of possible culprit actions is cleared since all transient norms are not violated any more.

If the algorithm passes the test $k = n$ and no action is marked as possible culprit, this means that the plan

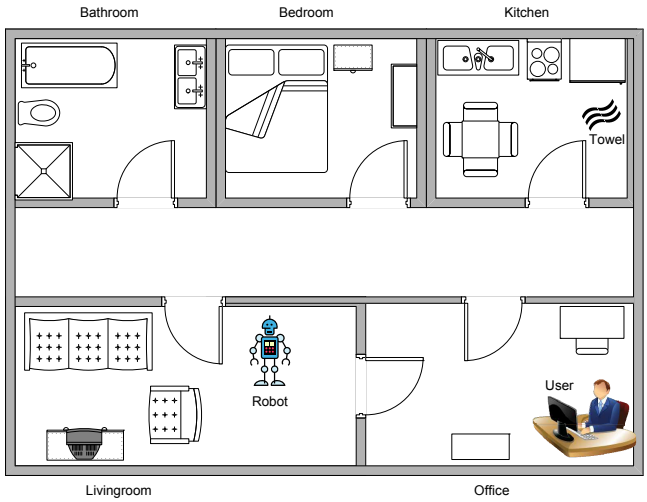

Fig. 5. A test environment that violates the normative constraints regarding the location of towels.

$(o_1, o_2, \ldots, o_n)$ does not violate any permanent norm, and that the final state does not violate any transient norm: hence, the plan is admissible and it can be executed by the robot.

5. AN ILLUSTRATIVE EXAMPLE

The above framework has been been implemented using the OWL-DL formalism (Dean et al., 2004) together with Pellet reasoner (Sirin et al., 2007) to deal with semantic knowledge in the T-Box, and traditional robot maps (Galindo et al., 2007) for the S-Box. We have used the Protegé ontology editor (Protégé Project, 2011) to create the contents of the T-Box, and the HTN-based planner JSHOP-2 (Ilghami and Nau, 2003) to generate plans. While each of these tools is very common in its respective domain, we emphasize that other tools could also be used without major changes in our approach.

We show a simple experiment whose main purpose is to clarify the working of our approach. Consider the apartment-like environment depicted in figure 5, and a T-box that represents two norms: a transient norm for the location of towels, i.e., towels should be in the bathroom, and a permanent norm related to the robot prohibition to be inside the office. The factual information for this example in the initial state is shown in figure 6.

Having this information stored at the T-box of the semantic map, the DL reasoner (Pellet) detects the violation of a normative relation, namely the position of the towel is not a bathroom yielding this explanation:

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

In order to fix this norm violation, the goal generator provides an initial goal by selecting an instance of the range of the given relation, i.e., an instance of the `Bathroom` concept. Thus, the goal `bring t1 b1` is proposed and the planning system produces the following solution:

```
s_i={(Towel t1), (Robot r1), (Kitchen k1),
(Bedroom be1), (Bathroom ba1) (Livingroom l1),
(Office o1) (Corridor c1), (located t1 k1),
(robot-location r1 l1), (nav l1 c1), (nav l1
o1), (nav o1 c1), (nav c1 ba1), (nav c1 be1),
(nav c1 k1)}
```

Fig. 6. Initial state. The norm violation is caused by the statement `(located t1 k1)`.

```
Loading ontology file:owl/goalgeneration.owl
Loading problem from file src\domain\facts.txt
Goal proposed: bring t1 b1
-------------------------
Plan
-------------------------
(!move l1 o1)
(!move o1 c1)
(!move c1 k1)
(!pickup t1 k1)
(!move k1 c1)
(!move c1 b1)
(!drop t1 b1)
--------------------
```

This plan will re-establish the consistency of the robot environment, i.e., at the end the towel (and also the robot) will be located in the bathroom. However, this solution is not admissible given that a permanent constraint is violated during the execution: the robot enters the office, which is not allowed by a permanent norm. Note that the transient norm regarding the location of the towel is violated too at intermediate states, but it is satisfied at the goal state.

The algorithm in figure 4 detects the expected violation of the permanent norm after analyzing the first action of the plan, i.e., `(!move l1 o1)`. Namely, the algorithm simulates the execution of this action by considering the definition of the `move` operator:

```
(:operator (!move ?x ?y)
   ((robot-location r1 ?x) (nav ?x ?y))
   ((robot-location r1 ?x))
   ((robot-location r1 ?y)))
```

and then adding to the semantic map the fact

```
(robot-location r1 o1)
```

obtained from the above operator binding `?x` to `l1` and `?y` to `o1`. This fact violates the permanent norm about the permitted locations for the robot. Therefore, the plan is rejected, the action `(!move l1 o1)` is disabled, and the planner re-invoked. [1] The planner then generates the alternative solution below:

```
Loading ontology file:owl/goalgeneration.owl
Loading problem from file src\domain\facts.txt
Goal proposed: bring t1 b1
-------------------------
Plan
-------------------------
(!move l1 c1)
(!move c1 k1)
```

[1] In our implementation, disabling of an action is done by deleting its preconditions from the initial state: in this case, `(nav l1 o1)`. This is an oversimplification, but it serves its purposes for this proof of concept implementation.

```
(!pickup t1 k1)
(!move k1 c1)
(!move c1 b1)
(!drop t1 b1)
--------------------
```

This solution is admissible since the permanent norm is met at every step in the plan and the transient one is met at the goal state, and it can therefore be passed to the robot for execution.

6. CONCLUSIONS AND FUTURE WORK

This paper extends our previous work on the utilization of semantic knowledge to enable a mobile robot to recognize and correct situations that do not comply with a given semantic model of the environment. The main novelty is a method that guarantees that the plans generated to solve a violation will not create other (transient or permanent) violations. Our development also revealed the necessity to distinguish between transient and permanent norms. The key message here is that "the end does not justify the means" in the sense that the robot should not keep the coherence between the perceived world and the established norms at any price, disobeying other norms at intermediate steps.

A distinctive feature of our approach to normative reasoning is that the normative model is provided in a declarative way, rather than by exhaustive violation-action rules. This allows the robot to resolve situations of conflict by reasoning about the cause of the problem and its possible solutions.

The work reported here addresses a complex problem, and many extensions can be considered. For instance, in our work we assume that the robot should *always* enforce consistency with the semantic knowledge. However, there are cases where norm violations might be allowed. For example, going back to our example, the robot could ignore the prohibition to enter the office if the user asks it to clean it during his absence. We speculate that our scheme can be extended to also cope with this and other exciting uses of semantic knowledge in mobile robots.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Government under contract DPI2011-25483, and by Örebro University's strategic funds.

This work was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine.

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., and Nardi, D. (eds.) (2007). *The Description Logic Handbook*. Cambridge University Press.

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

Brachman, R. and Levesque, H. (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann, San Francisco, CA, USA.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., and Stein, L.A. (2004). OWL web ontology language reference. W3C recommendation, W3C. URL `http://www.w3.org/TR/owl-ref/`.

Galindo, C., Fernandez-Madrigal, J., and Gonzalez, J. (2007). *Multiple Abstraction Hierarchies for Mobile Robot Operation in Large Environments*. Studies in Computational Intelligence, Vol. 68. Springer.

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

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

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann, San Francisco, CA, USA.

Ilghami, O. and Nau, D.S. (2003). A general approach to synthesize problem-specific planners. Technical Report CS-TR-4597 UMIACS-TR-2004-40, University of Maryland.

J. Hertzberg, A. Saffiotti (Eds.) (2008). Special issue on semantic knowledge in robotics. *Robotics and Autonomous Systems*, 56(11).

Kuiper, B. (1990). *Modeling Spatial Knowledge*, chapter Advances in Spatial Reasoning, Vol. 2, 171–198. The University of Chicago Press.

Meger, D., Forssen, P., Lai, K., Helmer, S., McCann, S., Southey, T., Baumann, M., Little, J., Lowe, D., and Dow, B. (2007). Curious george: An attentive semantic robot. In *IROS Workshop: From sensors to human spatial concepts*, 390–404.

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

Nüchter, A., Wulf, O., Lingemann, K., Hertzberg, J., Wagner, B., and Surmann, H. (2005). 3D mapping with semantic knowledge. In *RoboCup Int. Symp.*, 335–346.

Pronobis, A. (2011). *Semantic Mapping with Mobile Robots*. Ph.D. thesis, KTH Royal Institute of Technology, Stockholm, Sweden.

Protégé Project (2011). The Protégé ontology editor and knowledge acquisition system. URL `http://protege.stanford.edu/`.

Ranganathan, A. and Dellaert, F. (2007). Semantic modeling of places using objects. In *Robotics: Science and Systems Conf.*

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2), 51–53.

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