Robot Task Planning using Semantic Maps

Cipriano Galindo [a,*] Juan-Antonio Fernández-Madrigal [a]
Javier González [a] Alessandro Saffiotti [b]

[a] *Dept. of System Engineering and Automation, University of Málaga, Spain*
[b] *AASS Mobile Robotics Lab, Örebro University, Sweden*

Abstract

Task planning for mobile robots usually relies solely on spatial information and on shallow domain knowledge, like labels attached to objects and places. Although spatial information is necessary for performing basic robot operations (navigation and localization), the use of deeper domain knowledge is pivotal to endow a robot with higher degrees of autonomy and intelligence. In this paper, we focus on semantic knowledge, and show how this type of knowledge can be profitably used for robot task planning. We start by defining a specific type of *semantic maps*, which integrate hierarchical spatial information and semantic knowledge. We then proceed to describe how these semantic maps can improve task planning in two ways: extending the capabilities of the planner by reasoning about semantic information, and improving the planning efficiency in large domains. We show several experiments that demonstrate the effectiveness of our solutions in a domain involving robot navigation in a domestic environment.

Key words: Task planning, Robot maps, Mobile robotics, Knowledge representation, Cognitive robotics

1 Introduction

In an autonomous robot, task planning is used to plan a sequence of high-level actions that allows the robot to perform a given task. Task planning usually requires that several types of knowledge are encoded in the planner in a suitable way. These include causal knowledge, that is knowledge about the

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

Article published in Robotics and Autonomous Systems 56 (2008) 955–966

effects of the robot's actions, and world knowledge, that is knowledge about the objects in the world, their properties and their relations. For example, given the task to fetch a bottle of milk, a task planner might produce a sequence of actions like "Go to the kitchen, dock to the fridge, open it, perceptually acquire the milk bottle, grasp the bottle, close the fridge". To do so, the planner needs to know, e.g., that milk is stored in a fridge and fridges are in the kitchen.

Knowledge about the structure and the current state of the world is usually encoded in form of a *map*. The problem of how to represent, build, and maintain maps has been one of the most active areas of research in robotics in the last decades, and very valuable solutions to this problem are now available [1]. However, most of that work has focused on representations of the spatial structure of the environment, like metric maps [2,3], topological maps [4,5], or appearance-based maps [6]. This kind of maps are needed for navigation but they do not contain the more qualitative type of information needed to perform task planning. For instance, a metric map may represent the shape of a room, but it does not indicate whether this room is an office, a kitchen, or a bedroom – in fact, it does not even indicate that the given shape is a room. In most practical cases, this type of knowledge, which we call *semantic*, is encoded by hand in the domain description of the planner.

This tendency is now changing, and the field of autonomous robotics is witnessing an increasing interest in the so-called *semantic maps*, which integrate semantic domain knowledge into traditional robot maps [7–10]. The significance of these maps is that they can provide a mobile robot with deduction abilities (apart from basic skills like navigation, localization, etc.) to infer information from its world even when it has not been completely sensed. The use of semantic knowledge, thus, may enable a robot to perform in a more intelligent and autonomous manner. In the previous example, if the robot does not know where the kitchen is, but it has previously observed a microwave at a certain area, it can deduce that such an area is a kitchen, i.e., the place to go for accomplishing the "fetch a milk bottle" task.

In response to this tendency, a few recent works have addressed some issues related to the construction and usage of semantic maps [7,11–13] (see section 2). However, there are still open questions to be solved, including: how these maps can be automatically acquired, how semantic knowledge can be integrated with other types of knowledge in the maps (metric, topological, etc), and how it can be profitably used by the robot to plan and execute tasks.

This paper touches on all these questions, and focuses in particular on the last one: if a robot is endowed with an explicit representation of semantic information about its domain, how can a task planner profit from this ability? We explore this question in a constructive way. First, we propose our own *semantic map*, which integrates a spatial hierarchy of objects and places with

a semantic hierarchy of concepts and relations. The integration comes from linking elements from the spatial hierarchy to elements in the semantic hierarchy. We use our semantic map to engage in two case studies, according to two ways of using semantic information:

(1) *Deduce new information from the semantic structure, which is required for accomplishing task planning.* We consider different cases of this. First, to infer implicit properties of elements in the map that can be used by the planner: e.g., to deduce that a given area is a kitchen since the robot has perceived a fridge inside it. Second, to infer the existence of objects needed to solve a given goal: e.g., there must be a fridge in the kitchen, although this has not been observed, because kitchens usually contain a fridge. Third, to infer potential candidate objects to satisfy a goal, to be used to generate information gathering plans: e.g., to explore two rooms which have not been classified yet in order to determine which one is the kitchen.
(2) *Improve of planning efficiency by exploiting semantic constraints.* We use semantic knowledge to identify which categories are not needed for solving a given task, and to discard all the irrelevant instances before starting the planning process. For instance, if only the concepts `Kitchen`, `Living`, and `TV-set` are needed to achieve a certain goal, the planning system can ignore all the instances of the remaining concepts, thus improving its computational efficiency.

We complement our case studies by reporting two series of illustrative experiments. These experiments demonstrate that the use of semantic knowledge in task planning may endow a robotic system with increased ability to solve tasks with no human intervention (autonomy), to cope with situations that were not explicitly accounted for in its design (robustness), and to plan in large domains (efficiency).

Other works in the literature have included semantic knowledge inside robot system (see section 2). The work presented in this paper pushes this trend further by adding three main contributions: (1) a definition of a specific type of semantic map that links the traditions in robot maps and in knowledge representation; (2) an analysis of different uses of semantic maps for task planning; and (3) an experimental validation of the hypothesis that semantic maps can endow a robot with more autonomy, robustness and effectiveness.

The rest of this paper is organized as follows. In the next section we review some related works. Section 3 presents our own semantic map. Section 4 describes how semantic information complements spatial information for task planning. Section 5 presents the utility of semantics for improving task planning efficiency. Illustrative experiments of both case studies are presented in section 6, and finally section 7 concludes.

2 Related Work

World representation is probably one of the most important issues addressed in the robotics literature. Its paramount relevance stems from the necessity of having a proper model of the environment to enable a robot to plan, reason, and execute tasks. The use of world models that include semantic type of knowledge and inference capabilities dates back to the days of Shakey [14]. Flakey, a successor project of Shakey, used epistemic logics to reason about the beliefs, desires and intentions of robots and users [15]. More recently, Galindo et al. [7,16] and Zender et al. [13] have also used semantic knowledge, represented in some variant of desciption logics, to endow a robot with more general reasoning capabilities. The work presented here is one further step in this development.

In the more general domain of robot maps, a very large amount of work has been done focusing on the creation and use of maps specifically designed for robot navigation and localization. Most of these maps fall into two categories: *metric maps* [2,3], which represent geometric features of the environment, and *topological maps*, [4,5] which represent distinctive points and their topological relations. Given that both types of representations exhibit advantages and limitations, they can be combined into *hybrid maps* [17–20]. The inclusion of more high-level, semantic concepts in maps has traditionally received little attention in this development, although some important early works have been done by Kuipers [21] and by Chatila and Laumond [22].

Metric and topological maps are sufficient to provide a robot with its most basic functionality: navigate. The recent trend, however, strives for moving robots from research labs to human environments in applications like human assistance and entertainment [23,24], where robots are supposed to possess certain social skills apart from being able to navigate and self-localize. Social skills, for instance, require the ability to interact with humans using a human-like language which imposes that the robot must be endowed with a representation of the environment involving human concepts and their semantic relations.

Recently, the so-called *semantic maps* [12,13,25,26] have come up to capture the human point-of-view of robot environments, enabling high-level and more intelligent robot development and also human-robot interaction.

In order to maintain the usual abilities of the robot, i.e. navigation, semantic maps are usually combined with metric and/or topological representations into hybrid maps [7,13]. Some works [10,9] use the term semantic map" to refer to a spatial representation of the robot environment which also include the location and type of objects. The authors note that the presence of a given

at a given location may provide semantic information: e.g., a "printer room" is a space where a printer machine is located. However, these works focus on the problems of recognizing and locating objects in a map, and do not really exploit the potential that arises from semantic information.

Other works have considered semantic information for improving both human-robot interaction and the general performance and autonomy of mobile robots by inferring information from semantics. One of the forerunners of this research is [7], which presents a hybrid semantic map arranged into a two-hierarchical structure that enables a mobile robot to represent rooms and simple objects, e.g. a stove, and to infer implicit information. Subsequent works [11–13,26,27] have gone into this topic, developing more extensive semantic relations and robust mechanisms for the semi-automatic construction of semantic maps. However, none of these works exploits all the valuable possibilities that semantic information provides: they mostly use it to infer information about the acquired symbols according to their properties. This paper considers a similar approach for semantic mapping but presents a comprehensive study of its utility and utilization for robot task planning.

3 The Semantic Map

Our study presupposes that a mobile robot has enough abilities to perceive and map the environment as well as to access and exploit semantic knowledge about it. In this section, we describe how we represent and maintain this knowledge (spatial and semantic) in form of a semantic map.

3.1 Overview

Our semantic map comprises two separate 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, with special emphasis on spatial knowledge. The T-Box contains general semantic knowledge about that domain, giving meaning to the entities in the spatial box in terms of general concepts and relations.

This structure is reminiscent of the structure of hybrid knowledge representation (KR) systems [28], which are now 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

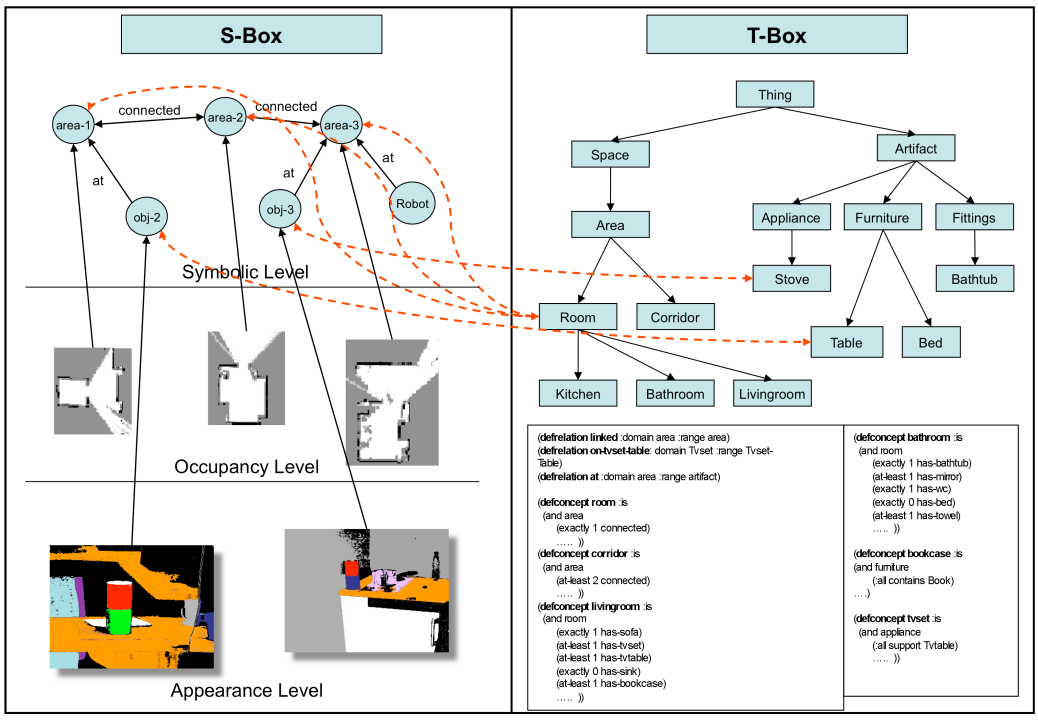

Fig. 1. An example of our semantic map. See explanation in the text.

goes one step further by extending the assertional component to be be more than a list of facts about individuals; it also associates these individuals to sensor-level information, and it is endowed with a spatial structure — hence the name S-Box. The representations used in the S-Box borrows from the rich tradition of map representation in robotics [1].

The rationale of our semantic map resembles the one of other recent approaches the robotic literature [11–13], namely, to link objects to semantic labels that are part of a full ontology in a KR system, which specifies the relation between semantic categories, and which can be used for reasoning.

Figure 1 shows a simple example of our semantic map. 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 semantic knowledge represented in it is a taxonomy. This taxonomy usually has a lattice structure, allowing multiple inheritance, although the one in Figure 1 has a simple tree structure. For the S-Box, the use of a hierarchical spatial representation is a convenient and common choice in the robotic literature [29–32] for dealing 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.

3.2 The spatial knowledge

The S-Box represents factual spatial information about the robot environment, including morphology of the space, position and geometry of objects, position of specific places of interest, sensor signatures perceived from those places, connectivity among places, and so on. We organize this information hierarchically with different types of knowledge at different levels.

The ground level of the spatial hierarchy, called *appearance level*, contains sensor signatures perceived from the environment plus information about the robot position from where the sensor information is gathered. The sensor signatures considered in our work are images of recognized objects and laser scans of distinctive places (of which there is none in this example). The *occupancy level* represents the free space in the environment, partitioned into bounded areas that correspond to rooms and corridors. In our experiments, these are built from laser data: Section 6.2 explains how this is done.

Elements in these two levels can be connected by links to indicate the area where a sensor signature is perceived. They can also be linked to the T-Box, if the mapping or perception processes are able to classify the spatial entities according to some predefined categories, e.g. rooms, places, and objects. In our system, the mapping process is only able to perform a very basic classification between places and objects. Objects are further classified as sofa, bed, table, cup, and so on. Places are classified as rooms and corridor depending on their eccentricity. This classification, that can be seen as part of an anchoring process [33,34], is the fundamental point in the connection between spatial and semantic knowledge. The assumption that the mapping process is able to classify the entities in the map is not restrictive: in the extreme case of a mapper with no classification ability, all entities can be linked to top-level concept "Thing", although this would of course limit the benefits of using semantic knowledge.

The upper level of the spatial hierarchy is the *symbolic level*, that maintains a symbolic representation of the space on which task planning is performed. This level contains a graph that represents the percepts stored at the lower levels (nodes) and their relations (edges). Nodes are directly created from percepts, while different types of edges that model different relations, like "connected" and "at", are created according to the geometrical information of the percepts.

3.3 The semantic knowledge

The semantic knowledge stored in the T-Box entails general knowledge about the types of the entities in the domain, and how they are related. These are

respectively represented in terms of *concepts* and *relations* in the tradition of description logics in KR [28]. Concepts and relations are structured into a hierarchy, often called an ontology, which provides an abstract description of the entities in the domain, and gives meaning to the terms used in the S-Box. For example, the perceptual signature of an object denoted by `area-2` can be associated to the concept `Kitchen` in the ontology. In addition, the T-Box supports inference by exploiting the structure of the ontology. For instance, if the ontology represents the fact that `Kitchen` is a specialization of the concept `Room` and that any `Room` has at least one door, we can infer that `area-2` has a door — although this door has not been explicitly asserted or observed yet.

In practice, we use the LOOM knowledge representation system to represent semantic knowledge. In LOOM, concepts are defined in relation to other concepts using the primitive `defconcept`. For instance,

```
(defconcept Kitchen :is (:and Room (:some has-appliance Stove)))
```

defines a kitchen to be a type of room characterized by having at least one appliance of class `Stove`. The definition

```
(defconcept Office :is (:and Room (:all occupant Employee)))
```

defines an office as a room whose occupants are all employees. LOOM provides two primitives to assert and query knowledge: `tell` and `ask`. In addition, LOOM provides a number of other functions to update or query the knowledge base: for example, the `retrieve` primitive can be used to find instances that belong to a given concept or that satisfy a given formula, like in

```
(retrieve ?x (:about ?x Office (:at-least 2 occupant)))
```

which will return all the instances that can be inferred to belong to the class `Office` and to have at least two persons in the role of `occupant`.

Figure 1 shows a fragment of the LOOM ontology used for the experiments in this paper.

3.4 Integrating spatial and semantic knowledge

The basis for the integration of the spatial factual knowledge contained in the S-Box and the semantic general knowledge contained in the T-Box is given by the links between the named entities in the S-Box and the concepts and relations in the T-Box. These links are created during the acquisition of the S-Box by exploiting its classification ability: whenever a new entity of a given class is created in the S-Box, it is given a unique name and this name is asserted in LOOM to belong to that class. For instance, suppose that, during

map building a new local gridmap is created in the S-Box. This grid would be called, say, `area-6`, and would be linked to the concept `Area` by the LOOM call `(tell (Area area-6))`. In Figure 1 the links so created are indicated by the red dotted lines.

These links alone, however, simply associate labels to entities in the S-Box. The real power of the semantic map comes from the fact that these labels are connected in a full ontology of the domain, and that this ontology can be used to infer new properties of the entities in the S-Box. In practice, the symbolic information in the semantic map is accessed by posting queries in LOOM, which perform inferences based on both the assertional knowledge in the S-Box and the semantic knowledge in the T-Box. For instance, to know the instances of the concept `Kitchen`, we issue the LOOM query

```
(retrieve ?x (Kitchen ?x))
```

Notice that there is no area in the map which is explicitly linked to the concept `Kitchen`, since the map builder does not have the ability to discriminate kitchens. LOOM, however, returns `(area-3)` as answer to the above query. In fact, `area-3` has been asserted to be a `Room`, and since an object of the class `Stove` has been observed `at` this room, `area-3` can be classified as an instance of the class `Kitchen`. In the next section we will see more examples of facts that can be derived by reasoning in the T-Box.

In the reverse direction, the synergies between the T-Box and the S-Box allow the robot to ground the semantic symbols in actual sensor-based entities that can be used for navigation. Continuing the example above, if the robot is given the task, in human-like terms, to "go to the kitchen", then the fact that the corresponding symbol `area-3` (which was deduced to be an instance of a kitchen) is linked to an occupancy grid allows the robot to give perceptual meaning to it, which can be used by the navigation routines.

4 Using Semantics to Improve Planning Capabilities

We now begin to explore the potential uses of the semantic knowledge incorporated in our semantic maps. The first class of uses that we study concern the ability to infer new information, which is entailed by semantic knowledge, in order to allow a robot to plan for a larger set of tasks than it would otherwise be possible. In all cases considered below, the key strategy consists in enlarging the initial state provided to the planner by including some of the facts that can be derived using semantic knowledge. In this section we do not assume any specific planner, although for the cases that involve partial observability we do assume that the planner is able to deal with incomplete states and with observation actions.

4.1 Explicitating implicit knowledge

Most task planners maintain an internal state, and compute how this state would change when different actions are applied [35]. In the simplest case this is a single propositional account of the state of the world, but more complex planners can work with multiple possible states and some even associate uncertainty measures to them [36,37]. In all cases, the initial state should contain enough information to allow the planner to find a way to achieve the goal: lack of information in the initial state may prevent the system from finding a plan even if one exists. In a robot system, the initial state is usually constructed by a combination of prior information and observations, for example: `(Room area1), (Stove obj1), (at obj1 area1), (at robot area1)`.

Semantic knowledge can be useful for enlarging and completing this initial state, by including into the S-box facts that can be inferred from the existing ones using the information in the T-box. For instance, given the definition of `Kitchen` in the previous section, the fact `(Kitchen area1)` could be deduced from the above facts and then added to the initial state. In this way, a given symbol in the S-box (`area1`) that was classified in a general way by the mapping process (`Room`) can be classified as an instance of a more specific class (`Kitchen`). This allows the planner to find a plan for a goal like "Go to the kitchen" which could not be solved from the original initial state.

Enriching the initial state with facts deduced through semantic knowledge has a back side: computing these facts may be costly, and a large initial state may reduce the efficiency of the planning process. The question, then, is how many of the facts that can be deduced should be included in the initial state. In general, we bound the amount of deduced information to be added to the initial state by providing a *context*: a set of concepts and relations which are considered relevant to the task. Algorithm 1 provides a way to compute the deductive closure for a given context. Steps 1 and 4 rely on the knowledge representation system underlying the T-box to retrieve all the instances that can be inferred to belong to the relevant concepts or relations: any system based on description logics provides primitives to do so. Steps 2 and 5 adds the inferred facts to the initial state used by the planner.

The possible contexts range between two extremes. At one end is the context consisting in the top concept and the top relation for the given ontology. Using this context, algorithm 1 would complete the initial state with the entire deductive closure, through the semantic knowledge base, of the known facts. At the other end is the context that only includes the concepts and relations that explicitly appear in the goal formula. This is the option that we have used in our experiments. For instance, given the goal

```
(exists (?x) (and (Kitchen ?x) (at robot ?x)))
```

Algorithm 1 SemanticClosure(C, R)

Require: A set of concepts C and a set of relations R
Ensure: Create all deduced ground facts about those concepts and relations
1: **for** $c \in C, x \in \{instances\ of\ c\}$ **do**
2: Add $c(x)$ to the internal state
3: **end for**
4: **for** $r \in R, \vec{x} \in \{tuples\ of\ r\}$ **do**
5: Add $r(\vec{x})$ to the initial state
6: **end for**

we first complete the initial state by finding all the instances that can be deduced to belong to the class `Kitchen` or to be in the relation `at`, and then invoke the planner on this expanded initial state. The exploration of other contexts is a subject for future work.

4.2 Inferring the existence of instances

Algorithm 1 expands the initial state by inferring new properties for the instances in the S-Box. Semantic knowledge can sometimes tell us even more than this: it can tell us that an object exists even if no instance is present in the S-Box, i.e., it has not been observed yet. Consider again our domestic robot, and suppose that it is given the task to approach the `tv-set`. If the robot has not seen any tv-set in the home yet, there will not be any instance of it in the S-box, and the planner will not find a plan to achieve the goal. Suppose, however, that the T-box contains the following definition of the class `LivingRoom`:

```
(defconcept LivingRoom :is (:and Room (:some at TvSet) ...))
```

meaning that living rooms are rooms that have (at least) a tv set. Then, if the robot knows that `area3` is a living room, it should be able to infer that there is a tv-set, although this has not been observed yet, and that the tv-set is located at `area3`.

In order to account for non-observed objects whose existence can be inferred from semantic knowledge, we create specific Skolem-type constants that represent witnesses for those objects, and we expand the initial state of the planner by adding the relevant assertions that concern those instances. In our example, we create a new constant `TVSET-0` to represent the non-observed tv-set, and add the following facts to the initial state: `(TvSet TVSET-0)`, `(at area2 TVSET-0)`. From this expanded initial state, the planner can find a plan for the goal 'approach the tv-set' provided that it can find a way to reach `area3`. The above procedure is described in general terms in Algorithm 2. The core of the algorithm is step 2, that looks for an object x that can provably be

Algorithm 2 AssertWitness(C)

Require: A concept C for which no instance has been asserted yet
Ensure: Create witness instances for C if they exist

```
for R ∈ {known relations in T-box} do
    for x ∈ {instances such that: ∃y. (R(x, y) ∧ C(y))} do
        Create new Skolem constant k_C^x
        Add C(k_C^x) to the initial state
        Add R(x, k_C^x) to the initial state
    end for
end for
```

in relation with at least one instance of C, through some relation R. If such an object exists, then also an instance of C must exist, and the algorithm adds the corresponding facts. In our example, this step yields for $x =$ `area3` and $R =$ `at`. Algorithm 2 is intended to be called after Algorithm 1 on those concepts C for which no known instance was found in Algorithm 1. Concrete examples involving both algorithms will be shown in section 6.

In general, the planner should take into account that the instances introduced by algorithm 2 do not represent actually perceived objects. A simple way to do so is to tag these instances by a special predicate, e.g., `Virtual`, and to extend the precondition of the robot's actions to require that the action target is not virtual. We then need to endow the robot with observation actions that are expected to perceive the actual objects and hence remove the `Virtual` tags.

4.3 Dealing with partial observability

In some cases, instances cannot be classified because some of the properties that would determine their class have not been observed yet. Suppose for instance that the initial state only contains the facts

```
(Room area1) (Room area2) (LivingRoom area3),
```

that is, the robot has not yet collected any specific information about the contents of `area1` and `area2`. Suppose also that the robot is given the task to go to the laundry room, where the concept `LaudryRoom` is defined by:

```
(defconcept LaundryRoom :is (:and Room (:some at WashingMachine)))
```

meaning that laundry rooms are rooms that contain at least one washing machine. Algorithm 1 would not be able to find any instance of a laundry room, since no washing machine has been observed. Algorithm 2 would also not help in this case, since the T-box does not contain any information that entails the existence of a laundry room.

Algorithm 3 FindCandidateInstances(C)

Require: A concept C for which there is no instance.
Ensure: Find alternative states with an instance classified as C
1: Parent ← parent-concept(C)
2: Siblings ← children-concepts(Parent)
3: Candidates ← direct-instances(Parent)
4: **for** i in Candidates **do**
5: Add ALT$\{S(i) \mid S \in$ Siblings$\}$ to the initial state(s)
6: **end for**

Semantic knowlege, however, can still be of use in this case, by providing indications about which instances in the S-box *could* be classified as `LaundryRoom`, if more information is collected. These are all the instances of a superclass of `LaundryRoom` that have not been further classified yet.[1] In our example, `area1` and `area2` are both direct instances of the class `Room`, which is a superclass of `LaundryRoom`. If further facts were asserted, these instances could be classified to a more specific concept below `Room` in the T-box lattice, e.g. `LaundryRoom`. By contrast, `area3` is not a candidate instance since this is not a direct instance of `Room`, having been already classified as `LivingRoom`.

The information about candidate instances can be used by the task planner, provided that this is able to deal with multiple hypotheses and partial observability. From this information, the planner can generate a conditional plan to acquire more observations about each candidate object, until one is found to be an instance of the needed class. In our example, a plan of this type could be informally described as: first go to `area1` and acquire observations; if after this `area1` is classified as `LaundryRoom` we are done; if not, go to `area2` and acquire observations; if after this `area2` is classified as `LaundryRoom` we are done; if not, the goal cannot be achieved. In section 6.2 we shall see a concrete example of a similar situation.

Algorithm 3 formalizes our strategy to find candidate instances for a given concept C. The algorithm first finds all the direct instances i of the parent P of C (step 3). For each such instance, then, it asserts that i could potentially be associated to any one of the children of P, including C itself. The statement in step 5 involves the creation of multiple alternative states, one for each possible classification of instance i. In our example, when considering the candidate instance `area1`, this statement results in the creation of two alternative states: one including `(LivingRoom area1)`, and one including `(LaudryRoom area1)`. Once these alternative states are provided to the planner, the planner knows that there is one possible state in which the goal can be solved, and it can

[1] Strictly speaking, these are the only candidate instances only when the ontology in the T-box has a tree structure with disjoint descendants, like the one shown in Figure 1 above. If the ontology departs from this structure there might be other candidates, but the above strategy still provides a good heuristic.

find a suitable conditional plan. How this is done in depends on the specific planner: a concrete example will be given in section 6.2.

4.4 Inferring new goals

Until now we have described how semantic knowledge can be used to enrich the initial state of the planner, thus extending the set of achievable goals. We now consider how semantic knowledge can be used to provide a robotic system with the ability to automatic generate new goals. This aspect has not been tested in our experiments, but we find it intriguing enough to deserve a brief discussion, albeit at a purely inspirational level.

The basic idea is to introduce concepts in the T-Box whose definition should be interpreted in a normative rather than descriptive way. For instance, consider the following concept definition

```
(defconcept Towel :is
   (:and Normative Household (:the location Bathroom)))
```

that defines towels as households whose unique location is a bathroom. Under a normative reading, this tells us that if an instance of a towel is found in a location different from the bathroom, the robot should generate the goal to bring it back to the bathroom. The additional superconcept `Normative` identifies those concepts that should be interpreted in a normative way.

The above idea could be implemented in an algorithm that regularly loops through the instances of the class `Normative` to check is some of them violates its semantic constraints. If one does, the violated constraint is used to generate a corresponding goal. Suppose that the following facts are in the S-Box

```
(towel obj3) (located obj3 area3) (Kitchen area3)
```

The instance `obj3` is inconsistent with the above definition of `Towel`. Since the violated constraint is `(:the location Bathroom)`, the algorithm generates the goal G that this constraint be satisfied, that is, that the location of `obj3` is a bathroom:

$$G = \texttt{(exists (?x) (and (Bathroom ?x) (location obj3 ?x)))}.$$

It should be noted that most KR systems provide primitives to check the above type of consistency. Loom, for instance, automatically marks inconsistent instances as `:incoherent`. This usage of semantic knowlege would provide robots with a higher degree of autonomy, and could be part of a simple kind of motivational architecture [38,39].

5 Using Semantics to Improve Planning Efficiency

Previous sections have exploited semantic knowledge for deducing new information. We now show that semantic knowledge can also be exploited for improving planning efficiency, an issue that is rarely considered in robotics in spite of its relevance when a robot has to deal with large amounts of information.

In general, a planning process searches for a sequence of ordered actions that transforms the initial state into the goal one. This search combinatorially explodes depending on the number of propositions and actions. A number of techniques have been proposed to alleviate this problem by reducing the applicability of actions ([35,40]), but not too much attention has been paid to the possibility of reducing the size of the initial state [41,42]. In this section, we describe an approach to do so, leveraging the abstraction provided by semantic knowledge. Our approach relies on what we call *semantic-level planning*: in a nutshell, this consists in translating the task planning problem from the original domain to an more abstract domain, defined in terms of semantic entities in the T-Box, where planning is performed on concepts instead than on instances. The result of this semantic-level planning is then used to prune the initial state of the original domain, by discarding irrelevant instances.

5.1 Formalization of semantic planning

We define our planning process as a tuple $\langle P, A, S_0, G\rangle$, where P is a set of propositions in a language Σ, A is a set of actions, $S_0 \subset P$ is the initial state, and $G \subset P$ is the goal state.

Let $\Sigma = \langle P_n, L, C\rangle$ be a first-order language composed of a set of predicate names $P_n = \{predname_1, \ldots, predname_w\}$, a set of literal constants $L = \{l_1, \ldots, l_n\}$, and the set of concepts of the T-Box $C = \{c_1, \ldots, c_t\}$ (which may correspond to types in PDDL).

Let f_C be a total function

$$f_C : L \rightarrow C$$

that associates each literal constant to a concept (its class). Let then the set of ground literals be $P = \{p_1, \ldots, p_m\}$, where $p_i = (predname_x\ x_{i1} \ldots x_{ik})$ is a k-ary proposition involving constants. Similarly we define $P^* = \{p_1^*, \ldots, p_q^*\}$ as a set of concepts propositions, being $p_i^* = (predname_y\ c_{i1} \ldots c_{ik})$ a k-ary proposition involving classes.

We also define a total function f_P by:

$$\begin{aligned} &f_P : P \rightarrow P^* \\ &\forall p_i = (predname_x\ x_{i1} \dots x_{ik}), f_P(p_i) = p_j^* : \\ &p_j^* = (predname_x\ f_C(x_{i1}) \dots f_C(x_{ik})) \end{aligned} \tag{1}$$

We overload this function to also work on sets of propositions:

$$F_P : \{P\} \rightarrow \{P^*\} : \forall J \subset P, F_P(J) = \{f_P(j_i)\}, \forall j_i \in J.$$

Finally, given an action $A = \{Pre, Post\}$ described in general as two sets of propositions indicating the precondition that must hold to be applied and the postconditions that will modify the current state, [2] we define the function F_A as:

$$\begin{aligned} &F_A : \{P\} \rightarrow \{P^*\} \\ &\forall a = \{Pre, Post\}, F_A(a) = \{F_P(Pre), F_P(Post\} \end{aligned} \tag{2}$$

With these definitions we can transform a conventional planning problem into a semantic-level planning problem of the form $\langle F_P(P), F_A(A), F_P(S_0), F_P(G)\rangle$. The semantic-level plan obtained by solving this planning problem is composed of a sequence of ordered actions that applied to the initial state ($F_P(S_0)$) achieves the goal one ($F_P(G)$), involving general concepts from the T-Box instead of particular instances. The main benefits of this transformation are:

- In spite of the complexity of the considered planning domain, semantic planning is performed on an abstraction of the original initial state, and therefore it has a low computational cost;
- The obtained semantic-level plan contains the particular classes involved in the final solution, giving hints about the instances of the spatial map that can be ignored during the non-semantic process of planning, thus reducing the complexity of the original problem.

5.2 Use of semantic planning

The semantic-level planning returns a set of concepts that are needed for solving the task at hand. The categories not involved in the resultant plan provide

[2] For simplicity in this formalization, we limit the definition of planning actions to two sets of propositions (preconditions and postconditions).

Algorithm 4 Semantic-level Planning(G,S_0,A)

Require: A goal G given as a proposition (pred-name p_1 ... p_n), the initial state S_0 and the available actions A.

Ensure: Find a plan involving concepts that solves the abstraction of G, i.e. $F_P(G)$

1: $G^s \leftarrow F_P(G)$=$\{f_P(p_i), \forall p_i \in G^s\}$=$\{$(pred-name parent-concept(p_1) ... parent-concept(p_n))...$\}$

2: $S_0^s \leftarrow F_P(S_0)$=$\{f_P(p_i), \forall p_i \in S_0\}$

3: $A^s \leftarrow F_A(A)$=$\{F_P(Pre_a) \cup F_P(Post_a)\ \forall a \in A\}$

4: $Plan_s \leftarrow$ Perform-Planning(S_0^s,A^s,G^s)

a valuable hint about the world information that can be discarded. Thus, the search space of planning can be largely reduced by ignoring irrelevant elements of the domain before the spatial planning executes, thus reducing the computational effort.

Algorithm 4 transforms the planning domain from the S-Box into a abstract domain in which occurrences of instances are translated to their correspondent concepts. For instance the translation of the information stored in the S-Box:

```
(Kitchen area-1) (Passway area-2) (Livingroom area-3) (Bookcase bk-1)
(Book b-1) (is-connected area-1 area2) (is-connected area-2 area3)
(has-book bk-1 b-1)(on-bookcase b-1 bk-1)(Robot r-1) (at r-1 area-1)
```

considering the T-Box of previous examples: is translated into:

```
(Room Kitchen)(Room Livingroom)(Corridor Passway)
(is-connected Kitchen Passway)(is-connected Passway Livingroom)
(is-connected Livingroom Passway)(on-bookcase Book Bookcase)
(has-book Bookcase Book)(Intelligent-Machine Robot)
(at Intelligent-Machine Kitchen)
```

This abstract domain clusters sets of instances into particular categories, significatively reducing the state space and therefore the complexity of the planning problem. For this example, and considering the goal `G=(taken b-1)`, the algorithm 4 yields $Plan_s$ =(MOVE Kitchen Corridor) (MOVE Corridor Livingroom) (APPROACH Bookcase) (TAKE Book), where we used Metric-FF planner for step 4.

This semantic-level plan is not meant to be executed by the robot, but to be used for discarding all the information in the S-Box (instances and their relations) which in irrelevant for the task at hand. Algorithm 5 isolates from the initial state S_0, derived from the S-Box, the instances that are needed for solving the task at hand, yielding the subset $S_0^r \subseteq S_0$ which only contains instances (and their relations) of the concepts that appear in the semantic-level plan. In our example, spatial information from concepts like Appliance,

Algorithm 5 Discard Spatial Information($Plan_s$)

Require: A semantic plan $Plan_s$

Ensure: Find a reduced version of the initial state for planning in the S-Box

1: $C \leftarrow \{$*concept names that occur in* $Plan_s\}$
2: $S_0 \leftarrow$ *Initial state coming from the S-Box*
3: $R \leftarrow \{$*instances of concepts in* $C\}$
4: $S_0^r \leftarrow S_0 \cap R$

Fittings, Household, and so on, are ignored. From this reduced version of the initial state, then, we can run any planner to obtain the final ground-level plan.

Although the above approach may seem similar to planning with types, there are important differences. Semantic-level planning provides a higher categorization of world elements, that permits us to firstly have the concepts (types in the PDDL sense) needed for solving the task, and then to ignore irrelevant instances. The improvement achieved by our approach with respect to a typed-planner (Metric-FF), shown in section 6.3, clearly reveals the benefits of semantic-level planning. Moreover, after we have operated the above simplification of the initial state through semantic-level planning, any other heuristic can be used to speed up planning even more.

6 Experiments

We now demonstrate the utility of our semantic map in the different situations discussed in the previous two sections by describing some illustrative experiments. In these experiments, we consider a mobile robot in a home-like environment that is given a variety of tasks. We describe two separate suites of experiments. The first one illustrates the benefits of semantic knowledge for enriching the state of the planner. These experiments have been run on an physical robotic testbed, described below. The second suite, described in section 6.3, demonstrates the benefits of semantic knowledge for improving task planning efficiency in large domains, and has been run in simulation.

It should be noted that the goal of these experiments was not to validate a full robotic system, but to illustrate the specific techniques described in the previous sections. As such, the different experiments have been run separately, and some simplifications have been introduced to the parts that are not relevant to the main point of this paper, like perception and user interface.

Fig. 2. Two view of the experimental environment. Left: the robot Astrid in the living room facing a sofa. Right: the robot at the entrance of the kitchen. The colored markers used to detect a sofa, a tvset and a table are clearly visible.

6.1 Experimental setup

For the real-robot experiments, we used an Activemedia PeopleBot robot called Astrid, equipped with a PTZ color camera and a laser range finder. Astrid navigates using the Thinking Cap, a robot navigation system based on fuzzy logic [43], on the top of a Player server [44]. The robot was placed in a home-like environment constructed inside our University building in Örebro [45]. Since reliable object recognition is not the focus of this paper, we simplified the domain by tagging each relevant object with a colored marker. [3] A color-based segmentation algorithm [46] together with geometric constraints was used to recognize the markers, which uniquely identify classes of objects. Figure 2 shows two views of the environment.

The T-Box in the semantic map was implemented using Loom, a popular open-source KR system based on an early version of description logic [47]. Task planning was done using PTLplan, a temporal-logic progressive planner able to deal with partial observability and uncertainty [36]. The ontology in the T-Box and the domains in the planner were coded by hand in these experiments. Figure 1 in section 3 above shows a fragment of the ontology.

6.2 Improving planning capabilities

The first suite of experiments tests the ability to use the implicit knowledge contained into the T-Box to improve planning capabilities.

In the first experiment, Astrid explored the environment and built a corresponding semantic map as explained in section 3. Occupancy grids were built

[3] See [13] for an example in which similar object recognition tasks are performed in an unmodified environment.

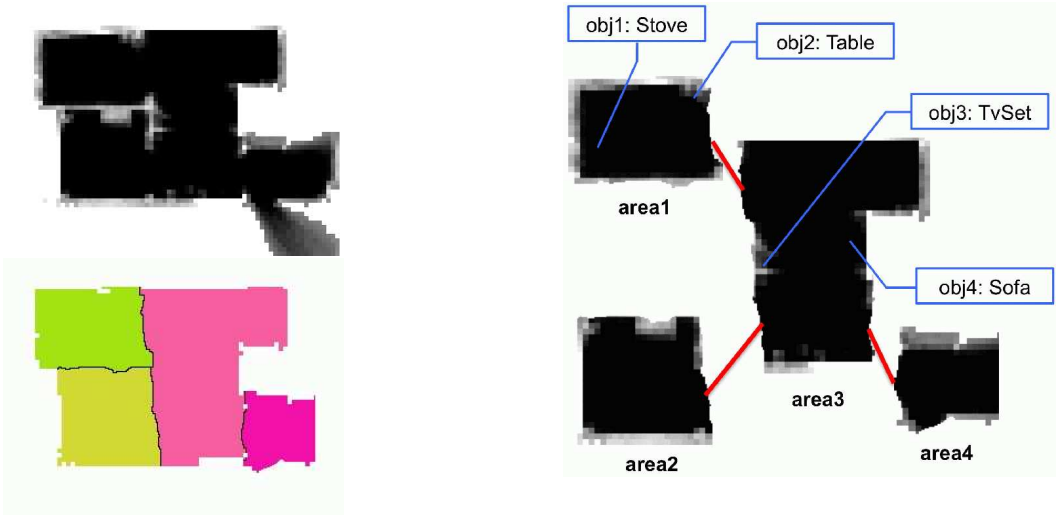

Fig. 3. The result of map building. Left: the occupancy grid (top) and its segmentation into areas (bottom). Right: the four local grids, their connectivity, and the position and classes of the detected objects.

from laser scans using the approach by Blanco and colleagues [48], that considers each scan as a node in a graph whose arcs measure the overlap between scans. The overall gridmap was then segmented into areas that correspond to bounded open spaces (rooms and corridors) using the technique described by Fabrizi and Saffiotti [49]. The segmentation algorithm also classifies areas into rooms and corridors based on their eccentricity, and extracts information about free-space connectivity between area. After segmentation, the map builder assigned a name to each area, and called LOOM to create the corresponding instances, to assign them to the general class `Room`, and to assert the appropriate connectivity relations:

```
(tell (Room area1) (Room area2) (Room area3) (Room area4))
(tell (conn area1 area3) (conn area2 area3) (conn area3 area4))
```

The vision system detected four known objects (markers) during exploration, which were were classified according to their color pattern. The position of each object relative to the robot was estimated from the observed size of the object, and it was converted to global coordinates to determine its location. For each detected object, then, the vision system created a name and asserted the corresponding facts in LOOM:

```
(tell (Stove obj1) (at obj1 area1)
      (Table obj2) (at obj2 area1)
      (TvSet obj3) (at obj3 area3)
      (Sofa  obj4) (at obj4 area3))
```

To keep the experiment simple, the robot was tele-operated and its self-localization was based on the Monte-Carlo localization system built-in inside Player [44]. Figure 3 shows the result of map building. The map built in this

experiment was used in all subsequent ones.

In the second experiment, the robot was standing in `area4` and it has been given the goal to go to the kitchen. All goals were given by typing a corresponding goal formula, using lisp-like syntax, which was then passed to the PTLplan planner. In our case:

```
(exists (?x) (and (Kitchen ?x) (at robot ?x)))
```

that is, the robot should be at a place which is classified as a being a kitchen. If PTLplan only looked in the spatial box, as in most existing systems, then it could not find any plan to satisfy this goal since no instance has been asserted to be of class `Kitchen` by the map builder. In our system, Algorithm 1 is first invoked with the concepts and relations appearing in the goal formula. In this experiment, Algorithm 1 was called with the single concept `Kitchen` as argument. It retrieved the relevant instances (step 1) using the LOOM call

```
(retrieve ?x (Kitchen ?x))
```

Since a stove was observed in `area1`, LOOM returned the answer `(area1)`. Algorithm 1 then asserted (step 2) the new fact `(tell (Kitchen area1))` into the S-Box. From this, PTLplan generated the following plan for the given goal, binding the variable `x` to `area1` and exploiting the connectivity relations: `((MOVE area4 area3) (MOVE area3 area1))`. This plan was sent to the ThinkingCap navigation system, that could execute it since the symbols `area4`, `area3` and `area1` denoted grounded entities in the S-Box.

In the third experiment, the robot was still standing at `area1` and it has been given the goal to go near the fridge:

```
(exists (?x) (and (Fridge ?x) (near robot ?x))).
```

No fridge was observed in the exploration phase, and hence there was no object in the S-Box which could be classified as fridge. Therefore, even after the execution of Algorithm 1, PTLplan could not find any object to be bound to the variable `?x`. At this point, Algorithm 2 was called with argument `Fridge` to check if the existence of a fridge could be deduced from the semantic knowledge in the T-Box.

In our implementation of Algorithm 2 we have restricted step 1 to only consider relations of a suitable type. In our domain, it is enough to look at the `at` relation if C is a subclass of `Artifact`, and at the `connected` relation if C is a subclass of `Space`. Step 2 has been implemented by the LOOM call

```
(retrieve (?x) (about ?x (:some R C)))
```

In our experiment, with `at` for R and `Fridge` for C, this calls returned `(room1)` since `room1` was known to be a kitchen and since kitchens have been defined to

have at least one fridge. A new Skolem constant FRIDGE-0 was then created, and the following facts were asserted in LOOM:

```
(tell (Fridge FRIDGE-0) (Virtual FRIDGE-0) (at FRIDGE-0 room1))
```

From this initial state, PTLplan generated the following conditional plan:

```
((MOVE area4 area3)
 (MOVE area3 area1)
 (OBSERVE area1)
 (COND
   ((Virtual FRIDGE-0 = F) (APPROACH FRIDGE-0) :SUCCESS)
   ((Virtual FRIDGE-0 = T) :FAIL)))
```

According to this plan, the robot navigates to the kitchen, observes it, and if FRIDGE-0 has been detected it approaches it. The OBSERVE action has been included by the planner because all the action operators that involve objects include the pre-condition (not (Virtual ?x)), and the (OBSERVE ?x) action has been declared to have the non-deterministic effect (not (Virtual ?x)). The OBSERVE action is intended to explore the room and visually acquire all the objects in it. Since its effect is non-deterministic, PTLplan has also included a conditional test on the fact that the fridge has been detected. [4] If this is not the case, the plan fails since, although the existence of a fridge can be inferred, no perceptually grounded entity for it can be created in the spatial hierarchy and therefore no data are available for navigation.

In our experiments, we have used a naive OBSERVE action that simply scans the room using a combination of robot rotation and camera motion. More complex observation strategies could be used, including the on-line generation of exploration plans [50], but the simple action above was enough for our goals.

In the last experiment, the robot was standing in area3 and it has been given the goal to go to the bedroom. The difficulty in this case is that no bed has been observed during the exploration phase, so no area could be classified as bedroom. Hence, Algorithm 1 did not result in any assertion about bedrooms. Moreover, there is no information in our semantic knowledge that implies the existence of a bedroom, so even the invocation of Algorithm 2 did not produce any instance of a bedroom.

In this case, we used Algorithm 3 to find instances that could possibly be classified as Bedroom if more information is acquired. The algorithm used LOOM functions to find that the parent concept of Bedroom is Room, and that its direct instances (candidates) are area2 and area1. These are areas

[4] Once could also use a simpler PDDL planner here, at the price of having to assume a deterministic OBSERVE action. This issue is independent on our use of semantic knowledge.

which have not been classified further, so they could be classified as any of the children of `Room`: `LivingRoom`, `Kitchen`, `Entrance`, or `Bedroom`. In step 5, the algorithm created a set of possible alternatives for the classification of `area2` and `area4`.

These alternatives have been given to PTLplan to create its initial state. PTLplan can deal with multiple hypotheses about the current state, which correspond to a situation of partial observability: the actual value of a variable has not been observed yet. In our case, the above alternatives were given to PTLplan in the following way:

```
(CLASS area1 = ((Bedroom 0.25) (LivingRoom 0.25) (Kitchen 0.25)
               (Entrance 0.25))
(CLASS area4 = ((Bedroom 0.25) (LivingRoom 0.25) (Kitchen 0.25)
               (Entrance 0.25))
```

where the numbers are probability values associated to each hypothesis (here assumed to be equiprobable). PTLplan can model observation actions which result in a reduction in the number of possible states, thus reducing uncertainty. Given the above initial state (plus the other known facts) PTLplan produces the following conditional plan for the goal to go to the bedroom:

```
((MOVE area3 area2)
 (OBSERVE area2)
 (COND
   ((CHECK-BEDROOM area2 = T) :SUCCESS)
   ((CHECK-BEDROOM area2 = F)
    (MOVE area2 area3)
    (MOVE area3 area4)
    (OBSERVE area4)
    (COND
      ((CHECK-BEDROOM area4 = T) :SUCCESS)
      ((CHECK-BEDROOM area4 = F) :FAIL)))))
```

This plan moves the robot to each candidate room in turn, performs an observation action, verifies if the room can now be classified as a bedroom, and if not it goes to the next one.

The `OBSERVE` action is implemented as in the previous experiment[5] . The predicate `(CHECK-BEDROOM x)` is implemented in the plan executor by a call to the LOOM subsystem: `(ask (Bedroom x))`. If the observation action has resulted in the observation of some discriminative elements (e.g., a bed), then

[5] A smarter strategy here would be to extract from the semantic knowledge base the distinctive elements to be observed in order to classify an area as a bedroom, e.g., a bed, and to use this information to parametrize the vision system or to only observe likely places for beds. This strategy was not attempted in our experiments.

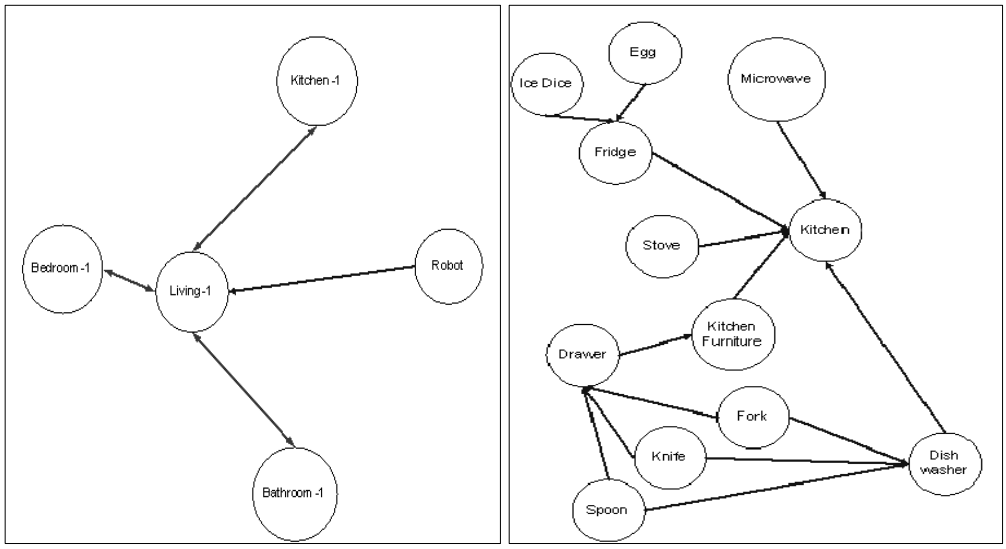

Fig. 4. Graph-based representation of the considered scenarios for simulations. Left: Part of the symbolic level of the spatial map of the simulated scenarios. Right: Semantic classes and their relations for the objects considered for a kitchen.

this call will succeed and the goal to be at a bedroom will be satisfied. Otherwise the plan will proceed to explore the other room.

6.3 Planning on a large domain

In order to test the use of semantics for improving task planning in large and/or complex scenarios we have performed a number of simulated experiments, using the Metric-FF planner [51] for generating both the semantic-level and the ground-level plans (recall section 5).

We have considered scenarios like the one graphically represented in Figure 4 simulating an apartment scenario with different rooms and their connections. Inside each room we consider a number of objects of different types and placements. For instance, Figure 4 (right) shows the objects to be considered in a kitchen: microwaves, fridges, forks, spoons, etc., which can be placed on tables, shelves, inside drawers, etc. In our simulations, we have generated different scenarios by randomly placing a number of objects (ranging from 100 to 5000) and intermediate places for navigation. For each scenario we consider five random "pick up an object" tasks by selecting one of the generated objects and an initial position for the robot.

In our experiments we compare the average planning time for the tasks under three different strategies: 1) Considering all the spatial information in a flat structure, 2) Considering all the spatial information hierarchically arranged into two levels (following [41]), and 3) executing algorithms 4 and 5 to reduce

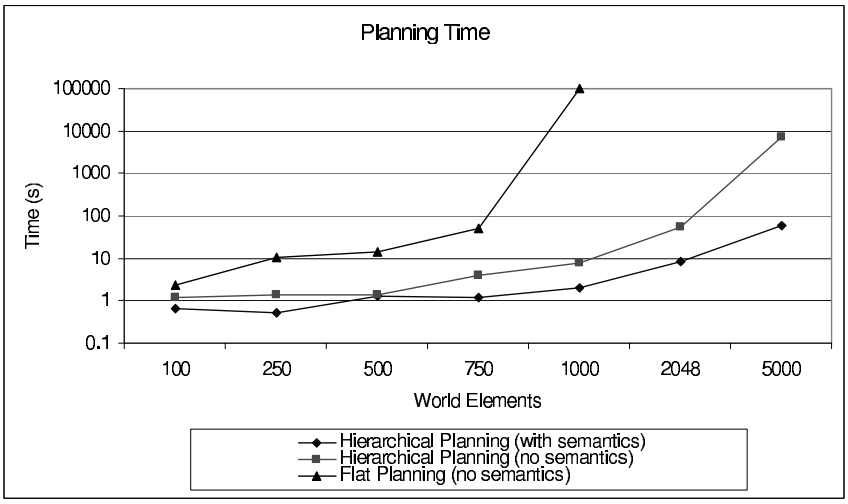

Fig. 5. Task Planning comparison. Average time for planning five random tasks within random variations of the complexity of the considered simulated environment.

the initial state before planning. It is important to remark that in the three cases we run the same planner (Metric-FF) and that the measured planning time includes the whole process, i.e., from when the goal is generated until the final plan is returned.

Figure 5 shows the average planning time for the set of five random tasks varying the complexity of the environment (number of elements) for the three commented cases. Although the behavior of all the planning strategies follows an exponential trend (which is usual in planning processes), Figure 5 clearly shows the benefits of using semantic information for planning. Also notice that in spite of the heuristics exploited by the Metric-FF planer, the reduction in the initial planning state based on the semantic information reduces the panning time, which proves that it actually alleviates the combinatorial explosion of the search involved in planning.

7 Conclusions

Semantic knowledge has an enormous potential to make robots more autonomous, by making better use of their available generic knowledge. The main intended contribution of this paper was to explore several different ways in which semantic knowledge can be used for one specific important problem in robotic: task planning. An secondary contribution is to define a well-founded semantic map that integrates spatial knowledge, modeled through the usual techniques found in the robotic literature, and common-sense (semantic) knowledge, modeled through approaches from the AI community. The proposed semantic map has been utilized to enable a mobile robot to plan and

execute tasks that could not be completed without the help of semantics.

Among the usages of semantic knowledge for task planning explored in this paper, we remark the ability to complete the current state with non-sensed information, the ability to automatically generate goals to maintain certain conditions, and the ability to improve planning efficiency. The last point in particular was achieved by performing planning on general concepts that abstract the current state. Our experiments have revealed the clear utility to include semantic knowledge in the task planning process of mobile robotics applications.

Finally, we emphasize that the semantic map described here can be seen as a general framework in which other spatial or knowledge representations can fit. Our short-term goal is test the usage of the semantic knowledge, as described in this paper, on more challenging domains, and to engage in a deeper investigation of the issue of using semantic knowledge as a source of motivational constraints to enable a robot to automatically generate its own goals.

Acknowledgments

This work was supported by the Swedish Knowledge Foundation and the Spanish Government (DPI2005-01391). We are grateful to Abdelbaki Bouguerra and Lars Karlsson for their useful comments and discussions.

References

[1] S. Thrun, Robotic mapping: A survey, in: G. Lakemeyer, B. Nebel (Eds.), Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

[2] M. Asada, Map building for a mobile robot from sensory data, IEEE Trans. on Systems, Man, and Cybernetics 37 (6) (1990) 1326–1336.

[3] S. Thrun, D. Fox, W. Burgard, Probabilistic mapping of an environment by a mobile robot, in: IEEE Int. Conf. on Robotics and Automation, 1998, pp. 1546–1551.

[4] H. Choset, K. Nagatani, Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization, IEEE Trans. on Robotics and Automation 17 (2) (2001) 125–137.

[5] B. Kuipers, Y. Byun, A qualitative approach to robot exploration and map-learning, in: Workshop on Spatial Reasoning and Multi-Sensor Fusion, Charles, IL, 1987, pp. 390–404.

[6] B. Kröse, O. Vlassis, R. Bunschoten, Y. Motomura, A probabilistic model for appearance-based robot localization, Image and Vision Computing 19 (6) (2001) 381–391.

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

[8] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, H. Surmann, 3D mapping with semantic knowledge, in: RoboCup Int. Symp., 2005, pp. 335–346.

[9] D. Meger, P. Forssen, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. Little, D. Lowe, B. Dow, Curious george: An attentive semantic robot, in: IROS Workshop: From sensors to human spatial concepts, 2007, pp. 390–404.

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

[11] G. Kruijff, H. Zender, P. Jensfelt, H. Christensen, Situated dialogue and spatial organization: What, where... and why?, Int. Journal of Advanced Robotic System 4 (2) (2007) 125–138.

[12] 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.

[13] H. Zender, P. Jensfelt, O. Martinez-Mozos, G.-J. Kruijff, W. Burgard, An integrated robotic system for spatial understanding and situated interaction in indoor environments, in: AAAI-07, Integrated Intell. Track, 2007, pp. 1584–1589.

[14] N. Nilsson, Shakey the robot, Tech. rep., No. 323, Artificial Intelligence Center, SRI International, Melon Park, CA (1984).

[15] M. Georgeff, A. Lansky, Reactive reasoning and planning, in: AAAI-87, 1987, pp. 677–682.

[16] C. Galindo, J. Fernandez-Madrigal, J. Gonzalez, A. Saffiotti, Using semantic information for improving efficiency of robot task planning, in: ICRA Workshop: Semantic Information in Robotics, 2007.

[17] 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.

[18] K. Kouzoubov, D. Austin, Hybrid Topological/Metric Approach to SLAM, in: ICRA, New Orleans (LA), USA, 2004, pp. 872–877.

[19] S. Thrun, Bücken, Integrating grid-based and topological maps for mobile robot navigation, in: 13th National Conf. on Artificial Intelligence, Portland, Oregon, 1996, pp. 944–951.

[20] N. Tomatis, I. Nourbakhsh, R. Siegwart, Hybrid simultaneous localization and map building: a natural integration of topological and metric, Robotics and Autonomous Systems 44 (2003) 3–14.

[21] B. Kuipers, Modeling spatial knowledge, Cognitive Science 2 (1978) 129–153.

[22] R. Chatila, J. Laumond, Position referencing and consistent world modeling for mobile robots, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 1985, pp. 138–145.

[23] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun, Experiences with an interactive museum tour-guide robot, Artificial Intelligence 114 (1–2) (2000) 11–18.

[24] R. Simpson, Smart wheelchairs: A literature review, Rehabilitation Research and Development 42 (4) (2005) 423–436.

[25] P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, B. Stankiewicz, Integrating Multiple Representatios of Spatial Knowledge for Mapping, Navigation, and Communication, Interaction Challenges for Intelligent Assistants, AAAI Spring Symposium Series, 2007.

[26] H. Zender, Learning spatial organization through situated dialogue, Master's thesis, Saarland University, Dept. of Computational Linguistics, Saarbrcken, Germany (2006).

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

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

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

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

[31] J. Fernandez-Madrigal, J. Gonzalez, Multi-Hierarchical Representation of Large-Scale Space, Int. Series on Microprocessor-based and Intell. Systems Eng., vol 24, Kluwer Academic Publishers, Netherlands, 2001.

[32] B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, H. Choset, The hierarchical atlas, IEEE Trans. on Robotics 21 (3) (2005) 473–481.

[33] S. Coradeschi, A. Saffiotti, An introduction to the anchoring problem, Robotics and Autonomous System 43 (2-3) (2003) 85–96.

[34] A. Sloman, J. Chappell, The altricial-precocial spectrum for robots, in: Proc of the Int Joint Conf on AI, 2005, pp. 1187–1192.

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

[36] L. Karlsson, Conditional progressive planning under uncertainty, in: Proc of the Int Joint Conf on Artificial Intell. (IJCAI), Seattle, USA, 2001, pp. 431–438.

[37] S. Majercik, M. Littman, Contingent planning under uncertainty via stochastic satisfiability, Artificial Intelligence 147 (2003) 119–162.

[38] A. Stoytchev, R. Arkin, Incorporating motivation in a hybrid robot architecture, Journal of Advanced Computational Intelligence and Intelligent Informatics 8 (3) (2004) 100–105.

[39] M. Quoy, P. Laroque, P. Gaussier, Learning and motivational couplings promote smarter behaviors of an animat in an unknown world, Robotics and Autonomous Systems 38 (3–4) (2002) 149–156.

[40] D. Bryce, S. Kambhampati, A tutorial on planning graph-based reachability heuristics, Assoc. for the Advancement of A.I. 24 (2007) 47–83.

[41] C. Galindo, J. Fernandez-Madrigal, J. Gonzalez, Hierarchical task planning through world abstraction, IEEE Trans. on Robotics 20 (4) (2004) 667–690.

[42] A. Botea, M. Enzenberger, M. Müller, J. Schaeffer, Macro-FF: Improving AI planning with automatically learned macro-operators, Journal of Artificial Intelligence Research 24 (2005) 581–621.

[43] A. Saffiotti, K. Konolige, E. Ruspini, A multivalued-logic approach to integrating planning and control, Artificial Intelligence 76 (1995) 481–526.

[44] T. Collett, B. MacDonald, B. Gerkey, Player 2.0: Toward a practical robot programming framework, in: Proceedings of the Australasian Conference on Robotics and Automation, 2005.

[45] The PEIS ecology project, Official web site, www.aass.oru.se/~peis/.

[46] Z. Wasik, A. Saffiotti, Robust color segmentation for the robocup domain, in: Int. Conf. on Pattern Recognition (ICPR), Quebec, CA, 2002, pp. 651–654.

[47] R. MacGregor, R. Bates, The loom knowledge representation language, Tech. rep., DTIC Research Report ADA183415 (1987).

[48] J. Blanco, J. Fernandez-Madrigal, J. Gonzalez, Towards a Unified Bayesian Approach to Hybrid Metric-Topological SLAM, IEEE Transactions on Robotics 24 (2) (2008) 259–270.

[49] E. Fabrizi, A. Saffiotti, Augmenting topology-based maps with geometric information, Robotics and Autonomous Systems 40 (2) (2002) 91–97.

[50] L. Karlsson, A. Bouguerra, M. Broxvall, S. Coradeschi, A. Saffiotti, To secure an anchor – a recovery planning approach to ambiguity in perceptual anchoring, AI Communications 21 (1) (2008) 1–14.

[51] J. Hoffmann, N. Bernhard, The FF planning system: Fast plan generation through heuristic search, J. of Artificial Intelligence Research 14 (2001) 253–302.