2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Semantic Knowledge-Based Execution Monitoring for Mobile Robots

Abdelbaki Bouguerra and Lars Karlsson and Alessandro Saffiotti
AASS Mobile Robotics Lab, Örebro University, SE-70182 Örebro, Sweden.
Email:{aba,lkn,asaffio}@aass.oru.se

Abstract— **We describe a novel intelligent execution monitoring approach for mobile robots acting in indoor environments such as offices and houses. Traditionally, monitoring execution in mobile robotics amounted to looking for discrepancies between the model-based predicted state of executing an action and the real world state as computed from sensing data. We propose to employ semantic knowledge as a source of information to monitor execution. The key idea is to compute implicit expectations, from semantic domain information, that can be observed at run time by the robot to make sure actions are executed correctly. We present the semantic knowledge representation formalism, and how semantic knowledge is used in monitoring. We also describe experiments run in an indoor environment using a real mobile robot.**

I. Introduction

Plan-based approaches have been a major trend in developing mobile robotic architectures capable of accomplishing complex tasks in non-structured environments. An important challenge that is faced by such architectures is how to carry out the execution of their plans so that tasks are achieved despite the presence of uncertainty and the dynamics of the real world. It has been recognized since the first days of autonomous mobile robots [1], that robust action execution requires monitoring the state of both the world and the robot to cope with contingencies that might occur at run-time.

Execution monitoring approaches have generally focused on using the explicit effects of actions to derive expectations that are compared with what is really produced by the execution of the action [2], [3]. This supposedly means that the effects to monitor are directly observable. That is, of course, not always realistic in a complex environment where checking expectations is a complex process. Therefore, we propose to use more advanced forms of reasoning in execution monitoring, where:

- high-level expectations are inferred from *semantic information*.
- these expectations are derived and verified online, for each step of the plan that is being executed.

By semantic information we mean knowledge about objects, their classes and how they are related to each other (this knowledge is sometimes called ”ontological” especially in the context of web contents). For instance, in an office environment, an office is a class whose individual instances (objects) denote rooms that have at least one desk and a chair; the entities desks and chairs are themselves defined as pieces of furniture . . . etc.

We use description logics [4] to encode semantic information. The major advantages of using description logics (DLs) are summarized as follows:

- DLs provide a concise representation of the world, as they can express general knowledge about classes of objects. Thus a lot of information can be kept implicit. For instance, one does not have to state explicitly that room-5, which is an office, contains a desk. Such information can be inferred from the general description of the class ’office’.
- DLs are fairly expressive yet supported by efficient inference mechanisms, making them practically useful.

As a concrete example of our approach, consider a mobile robot that is asked to deliver mail to the office of a certain person. As the robot enters the room asserted to be the office, it should expect to see at least a desk, a chair, and possibly a PC. These expectations are derived from the type of the room the robot is entering. If the robot is entering a kitchen instead, it should expect to see an oven, a sink,...etc. Therefore, checking semantic expectations when acting in indoor environments helps, among other things, to verify that the robot is in the correct room, and not (1) dislocated or (2) have an erroneous map.

Semantic information plays an important role in many application areas, and the semantic web [5] is just one example. There have also been applications of semantic information in mobile robotics, mainly for the purpose of facilitating human robot communication in a spoken language [6], classification of map spaces for navigation tasks [7], and as a means of publishing and sharing knowledge in multi-robot environments [8]. In fact semantic knowledge has been proved to be useful in building indoor 3D maps [9] where semantic information is expressed in terms of geometric constraints to classify points into floor, ceiling, or objects classes. However, to our knowledge this paper represents the first attempt to use semantic knowledge in the context of execution monitoring.

We should emphasize that our approach is intended as a *complement* to other approaches. We believe that execution monitoring needs to be performed at different levels, from detection of hardware errors to high-level deliberation. In this context, semantic knowledge can be one contributor to a multi-layered and multi-source monitoring process.

This work has been supported by the Swedish KK foundation and the Swedish research council.

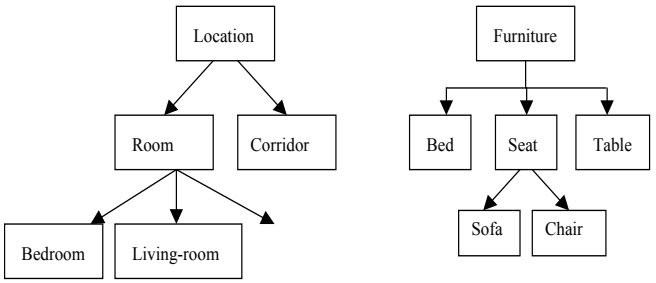

Fig. 1. Two taxonomies of some classes of objects in a house environment. (Left) the different types of locations. (Right) the different types of furniture. The relationships between the classes of the taxonomies are not shown

The rest of the paper is organized as follows. In section II, we describe how we encode and reason with semantic knowledge using the LOOM system [10]. In section III, we present the monitoring framework. Section IV presents a number of experiments intended to demonstrate the feasibility of the approach. They address monitoring navigation tasks, where the robot observes a room in order to determine that it is the type of room it expects to be in. Finally, before concluding the paper, we give an overview of related work.

II. SEMANTIC KNOWLEDGE

Semantic knowledge in this paper refers to the meaning of objects expressed in terms of their properties and relations to other objects. Objects that share the same properties and relations are grouped into classes (or concepts). For instance objects of type room and with at least one bed are instances of the class bedroom. While rooms with sofas and TV sets define a class of living-rooms,...etc.

To clarify our concepts, we use short examples of a house environment domain. The house will contain entities such as rooms (bathroom, living-room, kitchen,...), and furniture items (chairs, sofas, beds,...)(see figure 1).

A. Knowledge Representation

We opted for description logics (DL) [4] as a mechanism for representing the semantic domain information, since they provide a good trade-off between representation power and reasoning tractability. In fact one variant of the Semantic Web ontology language OWL uses descriptions logics as an inference engine.

1) Description Logics: are decidable fragments of first order logic intended as knowledge representation and management formalisms. They are used to represent domain knowledge of applications through the specification of the domain concepts and relationships between concepts (also called terminology). The description of the world consists in asserting properties and relations between individuals present in the domain. An important characteristic of DL is their reasoning capabilities of inferring implicit knowledge from the explicitly represented knowledge.

In DL formalisms, unary predicates represent concepts (also called classes) i.e. sets of individuals (also called objects), and binary predicates express relationships between individuals. Concept expressions can be built using a small set of connectives and constraints over the individuals that are in a relationship with a specific individual. Concepts that are not defined in terms of other concepts are called atomic concepts.

2) The LOOM System: In this paper, we use LOOM [10], a well established knowledge representation and reasoning system for modeling and managing semantic domain information, that uses description logics as its main inference engine. The choice of LOOM, was suggested by practical considerations: mainly because it is a well supported open source project. LOOM provides a definition language to write definitions of concepts and relations, and an assertion language to specify constraints on relations and concepts and to assert facts about individual objects.

Semantic knowledge in LOOM is organized in knowledge bases that contain definitions of concepts and relations between concepts. Concepts are used to specify the existence of classes of objects such as "there is a class of rooms" or "a bedroom is a room with at least one bed":

```
(defconcept Room)

(defconcept Bedroom
            :is (:and Room
                      (:at-least 1 Has-Bed)))
```

The term `Has-Bed` in the second definition specifies a relation between objects of class `Bedroom` and objects of class `Bed`. This relation is defined in LOOM as follows:

```
(defrelation Has-Bed
             :domain Bedroom
             :range  Bed)
```

The construct `(:at-least 1 Has-Bed)` specifies a constraint over the number of beds that can be in a bedroom. It is also possible to specify constraints over the types of objects an object can be in relation with. Note that `(:at-least 1 Has-Bed)` defines itself a class denoting all objects that have at least one bed.

More complex concept expressions are constructed by combining other concept names using a limited number of connectives (`and, or, not, implies`). The semantics of concept expressions are interpreted in terms of set theory operations (intersection, union,...) or in terms of equivalent first-order logic formulas over a non empty set of individuals.

Once the general semantic knowledge is constructed, specific instances of classes can be asserted to exist in the real world. For example:

```
(tell (Bedroom r1)(Has-Bed r1 b1))
```

Asserts that `r1` is an instance of `Bedroom` and results in classifying `b1` as a bed (because the range of the relation `Has-Bed` is of type `Bed`). The instance `r1` is also classified (deduced) automatically as an instance of the class `Room`.

Classification is performed based on the definitions of concepts and relations to create a domain-specific taxonomy (figure 1 shows a taxonomy of a house domain). The taxonomy is structured according to the superclass/subclass

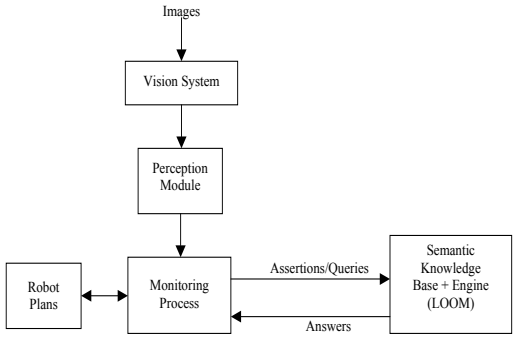

Fig. 2. The different modules involved in monitoring implicit expectations

relationships that exist between entities. When new instances of objects are asserted (added to the knowledge base), they are classified into that taxonomy.

III. Monitoring Framework

The general scenario involves a mobile robot acting in an indoor environment. The robot stores in its knowledge base (KB) information about its environment in terms of concepts (or classes of objects), relationships between concepts, as well as objects (individuals) expressed as instances of concepts. We will use the term *semantic information* to refer to such knowledge. The robot is supposed to execute tasks related to its environment, and to do so it formulates and executes symbolic plans. For example, if the robot is asked to clean the living-room, a plan to achieve such a task could be `(goto d1);(enter r1 d1);(clean r1)` where `r1` and `d1` are symbols denoting the living-room and a door that leads to it, respectively.

When executing action `(enter r1)`, a standard plan execution monitor would make sure that the robot is in `r1` simply by checking the current location provided by the robot's self-localization system. Our semantic-knowledge-based monitor checks also the implicit effects implied by being in `r1` through the use of semantic knowledge. In particular, since `r1` is an instance of the class `Living-room`, the robot looks for indications that it is in a living-room such as seeing a TV set or a sofa. If, on the other hand, the robot sees an oven, it should conclude that it is not in the living-room, since ovens must be in the kitchen. Such kind of information is derived from the semantics of the different rooms and objects present in the house.

A. Components

Figure 2 gives an overview of the different modules involved in monitoring task execution using semantic information. The architecture includes a vision system whose task is to provide a description of the scenes captured by the onboard camera in terms of percepts and their properties. This is mainly a description of objects of atomic classes like beds, sofas, and sinks.

The Perception module, on the other hand, establishes and maintains the correspondence between the percepts produced by the vision system and the symbols used by the symbolic planner and the semantic knowledge base. The robot plans module keeps track of the current plan and its execution context, as well as maintains the current expected state, which among other things includes the current expected location of the robot.

LOOM is used to store the semantic knowledge about the domain and object instances. It also processes and answers queries on its knowledge base. Finally, the monitoring process is responsible of making sure that plan actions are executed successfully, by comparing the expected consequences of an action to what has been perceived by the perception module.

B. The Monitoring Process

Figure 3 outlines the main steps performed by the semantic-knowledge-based monitoring process to check whether the execution of an action was successful. We will use examples of navigation actions to explain the steps of the process, however, the process is generic and can be adapted to check the execution of other types of actions.

The process gets as input the name of an individual that refers to an object of interest. In the case of monitoring a navigation action, the name of the individual refers to the expected location of the robot.

The pseudo operations prefixed by "LOOM" involve using semantic knowledge, whereas those prefixed by "PERCEPTION" involve the perception module.

semantic-monitoring(obj)
1. $CL \longleftarrow$ LOOM:get-class(obj)
2. $temp \longleftarrow$ PERCEPTION:perceived-object
3. $\Pi \longleftarrow$ PERCEPTION:perceived-prop&rel($temp$)
4. LOOM:create-instance ($temp, \Pi$)
5. **if** LOOM:is-instance-of($temp$, CL) **then**
6. success
7. **else if** LOOM:is-not-instance-of($temp$, CL) **then**
8. failure
9. **else**
10. - check implicit expectations
11. - lack of information

end

Fig. 3. The semantic-knowledge based monitoring process

In step 1, the process queries LOOM about the type of the object it is dealing with. For instance, if the object is the expected location `r1`, LOOM is queried about the asserted class of `r1`. Next, a temporary name is created for the perceived object that is of the same type as the object of interest obj. Then, the perception module is asked about the perceived properties and relations (to the other perceived objects) of the perceived object (step 3). The information returned by the perception module is used to create a new instance in the semantic knowledge base (step 4). For instance, if the perception module answers that the robot is in a room and that one chair `ch1` and one bed `b1` have been observed in that room, then the monitoring process

adds those facts to the semantic knowledge base through LOOM by issuing the following command:

```
(tell (Room temp)
      (Has-Chair temp ch1)
      (Has-Bed   temp b1))
```

Where `temp` is a temporary symbol used to refer to the current perceived room (where the robot is). LOOM performs an automatic classification of the newly created instance based on the properties and relations to the observed objects.

The next step is to check whether the classification is consistent with the type of the object *obj* (step 5). For our example, assuming that `r1` is a living-room, this is performed by sending the following two questions to LOOM:

```
(ask (Living-room temp))
(ask (:not (Living-room temp)))
```

LOOM's answers[1] to these two questions are interpreted in three ways:

1) Consistent Classification: The classification of the temporary object is the same as the object of interest *obj*. This is the result when all the implicit expectations are verified. Therefore, the monitoring module concludes that the action has been successfully executed (step 6). For our example, this happens when the robot has observed objects that cause the current location `temp` to be classified as a living-room (remember that that the robot is expected to be in room `r1` which is asserted to be a living-room).

2) Inconsistent Classification: This is the result when the classification of the temporary object is proved not to be of the same type as the object of interest *obj*. This occurs when an implicit expectation is violated. For instance, if the robot observes that it is in a room with a sink, then LOOM can confirm that the room cannot be a living-room, since the implicit expectation of living-rooms having zero sinks is violated. As a result of inconsistent classification, the monitoring module concludes that the action has failed (step 8). In our example this means that the robot is dislocated.

3) Unknown Outcome: This outcome results when there is lack of information which makes some constraints (implicit expectations) not known to hold nor to be violated. The constraints in question are those that appear in the definition of the class of the object of interest *obj*, which the monitoring process is asking about. In our example, if the constraint `(:at-least 1 Has-Sofa)` is not known to be true or false for the current room, then the room cannot be classified as a living-room by LOOM. Notice that the reason is that the robot did not see any sofa so far (lack of information).

In this case, the monitoring module may ask the semantic knowledge base whether the location is an instance of another class (that is not a superclass of the expected class) to check whether the robot is dislocated. If the class of the location is still not known, using perceptual information, the monitoring module assumes that the location is correct as long as no evidence of the contrary is detected. To enforce its decision each constraint (that is part of the definition of the class of the expected location) is individually checked looking for an evidence of being in that location (see the experiment in section IV-A).

[1] LOOM's answer to the first question is either YES or NO, where NO is interpreted both as FALSE and UNKNOWN. Therefore, the second question is used

C. Recovery Strategies for Navigation

In this section we give an overview of some possible recovery strategies for navigation actions. Whenever the monitoring module finds out that an action has not been executed successfully, a recovery procedure can be launched to correct the unexpected situation, either by updating the location of the robot or by collecting more information about the location where the robot is.

1) Location Update: This type of recovery is performed when the monitoring module concludes that the robot is dislocated (see section III-B.2 above). Consequently, the location of the robot should be updated to the right one. However, special care should be taken when performing location update, because sometimes the new location might be not unique. For instance, if all what the robot has observed so far is a sink, and sinks are defined to be either in a kitchen or a bathroom, then LOOM classifies the current sector as being a kitchen or a bathroom. Therefore the monitoring module should take this into consideration when notifying the localization module. In our experiments, the localization process creates a probability distribution over the possible locations to reflect this outcome (see the experiment in section IV-D).

2) Information Gathering : Information gathering can be performed to recover from situations where LOOM cannot determine the class of the location of the robot due to lack of information (section III-B.3 above). One way to gather information is to devise an active sensing plan whose aim is to move and make more observations in the room where the robot is located so that the implicit expectations not known to hold can be verified or refuted.

IV. Experiments

In order to validate our approach, we implemented the proposed monitoring framework on a Magellan Pro mobile robot called Pippi (figure 4) running a fuzzy behavior control architecture for navigation purposes [11]. The robot is equipped with 16 sonars, and a color CCD pan-tilt camera used by the vision system to recognize and classify objects. In practice, our vision system associates degrees of certainty to the possible shapes of an object. We let the perception module select the shape with the highest degree.

Since our main focus is showing the capacity of using semantic knowledge in monitoring and not on object recognition, we let simple shapes like balls and boxes stand in for beds, sofas, etc. The experiments reported below have been performed in a lab environment, placing the simple objects above to simulate pieces of furniture.

The following listing is a simplified and incomplete snapshot of the semantic knowledge base used in the experiments. The house comprises 5 rooms named `r1` to `r5`. `r1` is

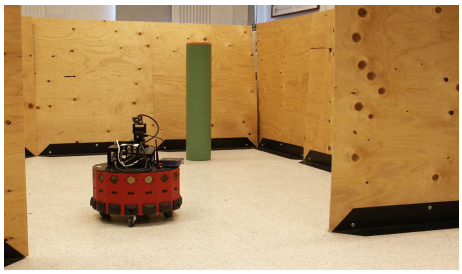

Fig. 4. Our Magellan Pro robot used in the experiments.

asserted to be a living-room, r2 is a kitchen, r3 and r4 are bedrooms, and r5 is a bathroom.

```
/* Relations */
(defrelation Has-Bed  :domain Bedroom  :range Bed)
(defrelation Has-Sink :domain (:or Bathroom Kitchen)
                      :range Sink)
(defrelation Has-Sofa :domain (:and Room (:not bathroom))
                      :range Sofa)
....

/*Atomic Concepts */
(defconcept Room)
(defconcept Sink)
...

/* Defined Concepts */
(defconcept Location :is-primitive (:one-of Room Corridor)
(defconcept Bed :is-primitive Furniture)

(defconcept Bedroom :is (:and Room (:at-least 1 Has-Bed)))
(defconcept Kitchen :is (:and Room
                              (:at-least 1 Has-Sink)
                              (:at-least 1 Has-Oven)))
(defconcept Living-Room :is (:and Room
                                  (:at-least 1 Has-Sofa)
                                  (:at-least 1 Has-TV)))
...

/* Assertions */
(tell (Living-Room r1) ) /* r1 is a living-room */
...
```

A. Successful Execution

The aim of this experiment is to show that the monitoring process can draw conclusions about the execution of an action in cases where not all implicit expectations are verified.

Here, Pippi executed the plan action (enter r1) to enter room r1. Once Pippi reached its destination location, the monitor module was called to check that the robot was inside the living-room using semantic knowledge. The monitoring module queried the perception module and found that two objects had been observed: one sofa and one table identified by the symbols sf1 and tl1) respectively. It tells LOOM about them:

```
(tell (Room temp)
      (Has-Table temp tl1)
      (Has-Sofa temp sf1))
```

Then it asked LOOM whether temp is an instance of the class Living-room. LOOM could not determine the class of temp based solely on those observations. The monitoring module, subsequently, decided to check if the robot was dislocated by asking LOOM if temp was an instance of another class (that is not a super class of Living-room). As the answer was negative and no evidence of not being in a living-room was detected, the implicit expectations (constraints) that could cause temp to be classified as a living-room were checked. This amounted to asking LOOM whether the constraints c1 = (at-least 1 has-sofa) and c2 = (at-least 1 has-TV) were satisfied by the instance temp. The reply of LOOM was that c1 is verified but c2 is not known to hold (since there no TV set had been observed so far).

The monitoring process decided that temp was a living-room, since there was nothing that conflicted with this classification. Consequently, the action (enter r1) was considered to have been executed successfully.

B. Failed Execution

In this experiment semantic knowledge is used to help the monitoring process to detect that the robot is dislocated. As in the previous experiment, Pippi was expected to be in r1 (the living room). But this time it saw a chair and an oven. The monitoring process asserted these facts and asked whether the current location was an instance of the class Living-room. LOOM answered that it did not know, because the implicit expectations of being in a living-room were neither verified nor violated. However, the current location was classified as a kitchen, as the domain of the relation Has-Oven is the class Kitchen.

As a result, the monitoring process concluded that the execution of the action (enter r1) had failed as the robot was in fact in r2, and not in r1. Therefore, the localization module was asked to correct the location, accordingly, and a recovery module was called to recover from such a failure by devising a navigation plan for the robot to navigate from its current location r2 to the living-room i.e. r1.

C. Uncertainty in Localization

As LOOM does not handle uncertainty, we devised a very simple ad-hoc procedure to take into account uncertainty in localization, where uncertainty about the location of the robot is modeled as a probability distribution over the possible locations of the robot. For instance, $P(\texttt{loc} = \texttt{r1}) = 0.8$, $P(\texttt{loc} = \texttt{r2}) = 0.2$ represents the belief that the robot is located either in r1 (with 80 % chance) or in r2 (with 20 % chance). In this case the location of the robot is a multi-hypothesis. Therefore the monitoring process is applied for each hypothesis individually.

To compute the posterior probability of being in a location loc, the prior probability of being in loc is scaled in accordance to the evidence that the current location of the robot is of the same type as the type of loc.

In this test run the localization module believes that the robot is either in r1 or r2 i.e. $P(\texttt{loc} = \texttt{r1}) = P(\texttt{loc} = \texttt{r2}) = 1/2$. Pippi then observes a sink; as seeing a sink is an evidence of being either in a kitchen or a bathroom, and given that the prior probability of being in bathroom is zero, the monitoring module concluded that the robot was in fact in the kitchen. Therefore, the posterior probability was set as $P(\texttt{loc} = \texttt{r1}) = 0$; $P(\texttt{loc} = \texttt{r2}) = 1$.

D. Self-Localization

The aim of this experiment is to show that the robot can use semantic information to determine in which location(s) it might be in. In this run, Pippi did not know which room it was located in, but it saw a bed in front of it. The Monitoring module told LOOM about this observation and asked LOOM about the types of the room where Pippi is. LOOM answered that the room was classified as a bedroom. Since there are two bedrooms in the house i.e. `r3` and `r4`, Pippi's belief about its location was set to being either in `r3` or `r4` with equal probability i.e. P(`robot-in = r3`) = P(`robot-in = r4`) = `0.5`.

V. Related Work

Although there is a considerable amount of work related to plan execution monitoring in mobile robotics, to the best of our knowledge, no research work has used semantic information to monitor plan execution. In the following we review some of the work done in the area, the reader is referred to [12] for a recent survey of the topic.

Reactive planning architectures, such as PRS [13] use hand-coded procedures to monitor the events that might affect the execution of plan actions. Consequently, expectations are explicitly coded in the monitoring procedure, which makes monitoring not flexible. In plan-based mobile-robotic architectures, such as Shakey [1], the LAAS architecture [14], and the work in [2], monitoring amounts to looking for discrepancies between the predicted state based on the explicit effects of actions, and the real world state as computed by the on-board sensing modalities. Other works include monitoring plan invariants [15] i.e environment conditions that have to be true during the execution of a plan.

The work presented in [7] focuses on connecting spatial information in maps to semantic information. Navigation tasks use semantic information to respond to human requests by inferring which spatial information of the map the robot should employ to achieve the task. The authors also briefly illustrate the use of semantic knowledge to detect some failures in navigation tasks, but they do not explore plan execution monitoring.

As the experiments presented in this paper concern the verification of expectations regarding the robot's location, one could, therefore think that our approach is related to the fundamental problem of self-localization. The relation is only coincidental, due to the use of navigation actions. Our approach could be used to monitor the execution of other types of actions (e.g. manipulation, and sensing) whose effects are not related to location.

VI. Conclusions

We have presented in this paper an intelligent plan-execution monitoring approach based on using semantic domain information to derive expectations that the robot can use to check the correct execution of its plan actions.

Our work has been implemented and validated using a real mobile robot. We would like to mention that the proposed approach is not dependent on any particular assumptions concerning the robotic architecture.

Finally, we want to emphasize that what is presented here is a first version of semantic monitoring and there is substantial potential to develop the method further, e.g. by working with other knowledge representation formalisms (including probabilistic ones). Unfortunately there is no workable description logic system which supports probabilistic reasoning (although some attempts have been made in that direction [16]). Therefore, we were restricted to encode only certain knowledge in our semantic knowledge base. Admittedly, that does limit the potential of the approach somewhat. But semantic knowledge can still add significant value to the monitoring process, as it has been demonstrated in this paper. In fact, we are currently investigating more elaborate strategies to combine semantic knowledge and uncertainty.

References

[1] R. E. Fikes, P. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artificial Intelligence*, vol. 3, no. 4, pp. 251–288, 1972.

[2] M. Fichtner, A. Großmann, and M. Thielscher, "Intelligent execution monitoring in dynamic environments," in *Proc. of the 18th Int. Conf. on Artificial Intelligence, Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating*, 2003.

[3] K. Z. Haigh and M. M. Veloso, "High-level planning and low-level execution: Towards a complete robotic agent," in *Proc. of the 1st Int. Conf. on Autonomous Agents*, 1997, pp. 363–370.

[4] "The description logic handbook: Theory, implementation, and applications," in *Description Logic Handbook*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003.

[5] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May, 2001.

[6] C. Theobalt, J. Bos, T. Chapman, A. Espinosa-Romero, M. Fraser, G. Hayes, E. Klein, T. Oka, and R. Reeve, "Talking to godot: Dialogue with a mobile robot," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2002, pp. 1338–1343.

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

[8] A. Chella, M. Cossentino, R. Pirrone, and A. Ruisi, "Modeling ontologies for robotic environments," in *Proc. of the 14th Int. Conf. on Software Engineering and Knowledge Engineering*, 2002, pp. 77–80.

[9] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann, "3d mapping with semantic knowledge." in *RoboCup International Symposium*, 2005, pp. 335–346.

[10] R. MacGregor, "Retrospective on loom," Information Sciences Institute, University of Southern California, Tech. Rep., 1999.

[11] A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multivalued logic approach to integrating planning and control," *Artif. Intell.*, vol. 76, no. 1-2, pp. 481–526, 1995.

[12] O. Pettersson, "Execution monitoring in robotics: A survey." *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[13] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An architecture for real-time reasoning and system control," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 7, no. 6, pp. 34–44, 1992.

[14] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[15] G. Fraser, G. Steinbauer, and F. Wotawa, "Plan execution in dynamic environments." in *Proc of the 18th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2005, pp. 208–217.

[16] D. Koller, A. Y. Levy, and A. Pfeffer, "P-classic: A tractable probablistic description logic," in *Proc. of the AAAI*, 1997, pp. 390–397.