Proc. of the International Conference on Intelligent Robots and Systems (IROS)
San Diego, California, October 2007

Handling Uncertainty in Semantic-Knowledge Based Execution Monitoring

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

Abstract— **Executing plans by mobile robots, in real world environments, faces the challenging issues of uncertainty and environment dynamics. Thus, execution monitoring is needed to verify that plan actions are executed as expected. Semantic domain-knowledge has lately been proposed as a source of information to derive and monitor implicit expectations of executing actions. For instance, when a robot moves into a room asserted to be an office, it would expect to see a desk and a chair. We propose to extend the semantic knowledge-based execution monitoring to take uncertainty in actions and sensing into account when verifying the expectations derived from semantic knowledge. We consider symbolic probabilistic action models, and show how semantic knowledge is used together with a probabilistic sensing model in the monitoring process of such actions. Our approach is illustrated by showing test scenarios run in an indoor environment using a mobile robot.**

I. Introduction

Plan execution and monitoring by mobile robots is a complex task as it involves dealing with uncertainty and unexpected situations. Uncertainty is an inherent feature of acting in real world environments, and it can cause even the best laid plans to fail. To cope with unexpected execution contingencies, plan execution systems employ monitoring procedures to make sure the execution of their plans do not divert from their intended course of action [1], [2].

Most plan execution monitoring approaches focus on using explicit effects of actions to derive expectations that are compared with what is really produced by the execution of the action [3],[4]. For instance, an explicit effect of grasping an object would be that the robot is holding the object, and the explicit effect of entering a room would be that the robot is inside that room. This supposedly means that the effects to monitor are directly observable. That is of course not always realistic in a real world environment where checking expectations is a complex process. Therefore, more advanced forms of reasoning involving semantic knowledge have been proposed to derive implicit expectations and monitor them online [5]. For example, a mobile robot moving into an office, should expect to see at least a desk, a chair, and possibly a PC. If the robot is entering a kitchen instead, it should expect to see an oven, a sink, etc. These implicit expectations are details that would add complexity to the planning task, if the task-planner has to reason about them. That is why they are encoded in a separate semantic knowledge base, i.e., outside the action descriptions the planner uses.

This work has been supported by the Swedish KK foundation.

Our research builds upon the work in [5], where monitoring was limited to actions with deterministic effects, i.e., having only one outcome, using a boolean treatment of expectations, i.e., evaluated to be either true, false or unknown. Our main contribution is the extension of the Semantic-Knowledge based Execution Monitoring process to take into account quantitative uncertainty in the form of probabilities in states, actions, sensing and the way we interpret expectations in our semantic knowledge. More specifically, we allow for actions to have different outcomes each with a probability of occurrence, and sensing to be unreliable. Consequently, we can assign a probability to whether a certain expectation is verified, such as "I'm in an office with 0.9 probability" – in the framework of [5], that would just have been a "unknown".

In our approach, the monitoring process uses semantic knowledge in the following way:

- For each possible outcome of the action being monitored, a set of implicit expectations are derived. For instance, if one outcome is to be in an office, the implicit expectations of having at least one desk and at least one chair are derived from the semantic domain-knowledge. If the second outcome of the action is ending up in the printing-room, the implicit expectations would include having at least a printer, one copier, etc.
- Those expectations are used to estimate a probability distribution over what we would expect the actual world state to be like. For instance, the expectations of having at least one desk implies that the probability of having no desk is zero, while the probability of having one, or two desks is greater than zero.

Besides uncertainty about the real world state, we consider uncertainty in sensing through a model that expresses the probability of what is observed for a given world state. In its general form, the sensing model permits:

- To state whether an object that exists in the real world is seen or not (false negatives), by taking, e.g., occlusion into account.
- How a seen object is classified, i.e., the model accounts for misclassification of objects when they are seen (false positives). For instance, a sofa may sometimes be mistaken to be an arm chair.

The monitoring module uses the prior probability distribution over the outcomes of the executed action together with the semantic knowledge-based probability estimates and the sensing model to compute the posterior probability of the outcomes. Thus, the monitoring task becomes more like a Bayesian belief update task.

It is important to emphasize that, despite the examples used in the paper, the proposed approach is not about self-localization, but about execution monitoring of high-level plan actions, which might include navigation actions as well as other types of actions.

Although there is a considerable amount of work related to plan execution monitoring in mobile robotics [4], to the best of our knowledge, no research work has used semantic knowledge to monitor the execution of actions with uncertain effects and noisy sensing. The work in [6] briefly illustrates the use of semantic knowledge to detect some failures in navigation tasks. Execution monitoring of stochastic actions has been addressed for low-level navigation in indoor environments [7], but without using semantic knowledge.

In most plan-based mobile-robotic architectures, such as Shakey [1], the LAAS architecture [8], and the work in [9], 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. Besides not using semantic knowledge, and not considering uncertainty, reactive planning architectures, such as PRS [10] 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 the next section we give an overview of the semantic knowledge-based execution monitoring framework proposed in [5]. Then, we show how monitoring is carried out taking into account noisy sensing and semantic knowledge. Before concluding, we describe test scenarios run on a mobile robot.

II. Semantic Knowledge-Based Monitoring

Semantic knowledge refers to 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, an office is a concept that denotes rooms that have at least one desk and a chair; the entities desks and chairs are themselves defined as pieces of furniture, etc.

We opted for description logics (DL) [11] to represent and manage the semantic domain knowledge, since they provide a good trade-off between representation power and reasoning tractability. An important characteristic of DL is their reasoning capabilities of inferring implicit knowledge from the explicitly represented knowledge.

A. The LOOM System

In our work, we use LOOM [12], 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 motivated by practical considerations: mainly because it is a well supported open source project. LOOM provides a 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-most 1 has-sofa)
                              (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 atomic construct `(at-least 1 has-bed)` above specifies a constraint over the number of beds that can be in a bedroom. Number constraints in LOOM are expressed by the constructs `(at-least n Rel)`,`(at-most n Rel)`, and `(exactly n Rel)`. They are used to encode constraints on the number of objects (`n`) of one class that are related to another object of another class through the relation `Rel`.

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 or in terms of equivalent first-order logic formulas over a non empty set of individuals.

Once the definitions are specified, specific individuals 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. The taxonomy is structured according to the superclass/subclass relationships that exist between entities. When new instances of objects are asserted (added to the knowledge base), they are classified into that taxonomy.

B. Previous Monitoring Framework

The approach proposed in [5] uses semantic knowledge to monitor the execution of actions with a single expected outcome. The key idea is to derive implicit expectations related to the successful execution of the action and check them using available perceptual information. For instance, executing the action `(enter r1)`, where `r1` is asserted to be a bedroom, involves checking whether at least one bed has been perceived in the current room, to conclude that the action has been executed successfully.

The process of checking the implicit expectations gives rise to one of the following three outcomes:

1) All expectations hold. Therefore, the action is considered to have been successfully executed.

2) At least one expectation is violated, thus the monitoring process concludes that the action has failed.
3) The truth values of some expectations are not known. Thus the monitoring process cannot tell whether the action has succeeded. This happens when there is missing information (for instance due to occlusion) that would be needed to evaluate the expectations.

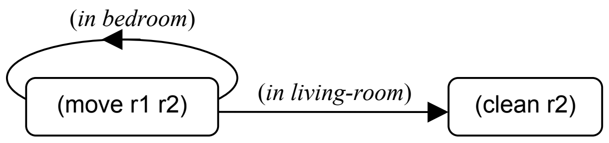
Note that checking an expectation gives a ternary answer: success, failure, or unknown. In addition, perceptual information is assumed to be reliable.

III. Execution Monitoring under Uncertainty

We extend the semantic knowledge-based execution monitoring to handle the issues of uncertainty in sensing and actions that have nondeterministic outcomes. For instance executing the action `(move r1 r2)` – to move from room `r1` to room `r2` – might result in one of two possible outcomes: either moving to `r2` or unintentionally remaining in `r1`. In our framework, we consider action models that assign probabilities to outcomes. For instance the first outcome of `(move r1 r2)` can occur with 0.8 probability, while the second outcome occurs with 0.2 probability.

Planning involving uncertainty in sensing and effects of actions is carried in the space of belief states, where a belief state is a probability distribution over possible states that share the same observation – see, e.g., [13].

We use a forward-chaining probabilistic conditional planner [14] that starts with an initial belief state and adds actions until a belief state, where the goal is satisfied, is reached. If the insertion of an action gives rise to a set of belief states (due to uncertainty in observations), the planner inserts a branch for each resulting belief state and continues planning for each branch separately. Therefore the resulting plan is conditional, where the condition of each branch is the observation of the corresponding belief state. For instance, the following plan could be generated to clean room `r2` which is asserted to be of type `living-room` starting from room `r1` (asserted to be of type `bedroom`), where nodes are labeled with the names of actions, while the labels of the arcs represent the conditions on which to branch.



After the execution of each action, the execution monitoring process is called to check whether the actual effects of the action execution are the same as the ones predicted by the planner (based on the action model). This is basically done through computing the posterior probability distribution of the outcomes using the execution-time acquired observations, i.e., the monitoring process computes the actual execution-time belief state.

More specifically, if $\mathbf{o}$ denotes the collected observations, then the posterior probability of the action resulting in a specific outcome r is computed using Bayes formula:

$$p(r|\mathbf{o}) = \frac{p(\mathbf{o}|r)p(r)}{p(\mathbf{o})} \tag{1}$$

To compute the posterior $p(r|\mathbf{o})$, two probabilities functions are needed: (1) the prior probability $p(r)$ which is given in the action model, and (2) the observation function $p(\mathbf{o}|r)$.

In the following we show how we compute the observation function $p(\mathbf{o}|r)$. We will use the following notation: boldface letters denote vectors, capitalized letters denote variables, and uncapitalized letters denote specific values of the variable denoted by the same letter but capitalized, e.g., o is the same as $O = o$ and $\mathbf{x}$ is the same as $\mathbf{X} = \mathbf{x}$. The i^{th} element of a vector $\mathbf{X}$ is denoted by X_i.

To compute the observation function $p(\mathbf{o}|r)$, we need the following:

- A random variable R whose domain values $\{1, \ldots, m\}$ denote the different action outcomes, with a prior probability $P(R = r)$ of R for each single outcome r.
- A set of N atomic concepts, defined in the semantic knowledge base, each representing a class of observable objects C_i. For example, C_1 = `bed`, C_2 = `sofa`, etc. We consider only atomic concepts whose numbers are constrained by a number constraint construct.
- A random vector $\mathbf{O}$ of size N such that its ith random variable O_i represents the number of observed objects of type C_i. For instance, if C_1 refers to the concept `bed`, O_1 represents the number of observed beds.
- A random vector $\mathbf{S}$ of size N such that its ith random variable S_i is a state variable whose values are the actual number of objects of type C_i. In our model, each state variable S_i depends directly only on R. Each state variable S_i takes values in a domain $V_i \subseteq \mathbb{N}$

Taking this into consideration, equation (1) becomes:

$$\begin{aligned} p(r|\mathbf{o}) &= \sum_{\mathbf{s}} p(r, \mathbf{s}|\mathbf{o}) \\ &= \alpha \sum_{\mathbf{s}} p(\mathbf{o}|\mathbf{s}) p(\mathbf{s}|r) p(r) \end{aligned} \tag{2}$$

Where $\mathbf{s}$ ranges over values belonging to $V_1 \times V_2 \times \cdots \times V_N$, and $\alpha = 1/p(\mathbf{o})$ is a normalizing factor. To compute the observation function, two probability mass functions are required: (1) A sensing function $p(\mathbf{o}|\mathbf{s})$ that describes the probability of observing $\mathbf{o}$ when the real world state is described by $\mathbf{s}$. (2) A state function $p(\mathbf{s}|r)$ that describes the probability of $\mathbf{s}$ when the outcome of the action is r.

Although we focus on constraints on the number of objects of different types we want to emphasize that our approach is not limited to that. One could have random variables representing features such as color or size with a set of different values, and constraints over these values, e.g., `color` $\in \{$ `red, yellow, white`$\}$.

A. The Sensing Model

We start by deriving a general, but computationally expensive, sensing model; a simpler version is provided further down.

1) General Sensing Model: The sensing function $p(\mathbf{o}|\mathbf{s})$ represents the sensing model of the robot. The term $p(\mathbf{o}|\mathbf{s})$ specifies that given certain actual values $\mathbf{s}$ for the state variables, what is the probability that we will observe o_1 objects of type C_1, o_2 objects of type C_2,$\cdots$, and o_N objects of type C_N? In its general form $p(\mathbf{o}|\mathbf{s})$ permits:

1) To state if an object is seen or not (false negatives), and
2) How a seen object is classified, i.e., the model accounts for misclassification of objects when they are seen (false positives).

As all random variables in $\mathbf{O}$ and $\mathbf{S}$ depend on each other, there are an exponential number of probabilities $p(\mathbf{o}|\mathbf{s})$ that require to be specified. We break this dependency by introducing N random vectors $\mathbf{G}_{i:1\leq i\leq N}$ (each of dimension $N+1$). Each $\mathbf{G}_i$ depends directly only on the ith state variable S_i and $p(\mathbf{g}_i|s_i)$ expresses the probability of classifying s_i objects of type C_i as $g_{ik} (k = 1 \dots N)$ objects of class C_k. The number of missed (unseen) objects of type C_i is denoted by $g_{i(N+1)}$.

$p(\mathbf{g}_i|s_i)$ is, therefore, a probability mass function of a multinomial whose parameters are $n = s_i$ and $p_{k(1\leq k\leq N)}$ is the probability of classifying an object of type C_i as of type C_k, and p_{N+1} is the probability of missing (not seeing) an object of type C_i. Figure 1 shows the dependency structure of $R, \mathbf{S}$, $\mathbf{O}$, and $\mathbf{G_i}$.

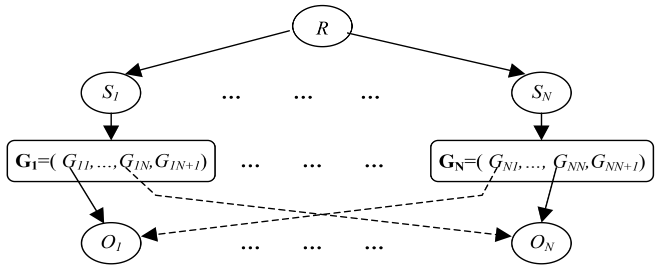


Fig. 1. The dependency structure of the different random variables used in the state and sensing functions.

Note that $o_i = \sum_{k=1,N} g_{ki}$, i.e., o_i represents the total number of objects classified as of type C_i; either correctly, i.e., g_{ii} or incorrectly, i.e., g_{ki} (for $k \neq i$). For example, the number of observed chairs is the total number of objects classified (correctly or incorrectly) as chairs. Thus we have

$$p(o_i|g_{1i},\dots,g_{Ni}) = \begin{cases} 1 & \text{if} \sum_{k=1,N} g_{ki} = o_i \\ 0 & \text{otherwise} \end{cases}$$

Finally, the general sensing model is formulated as:

$$\begin{aligned} p(\mathbf{o}|\mathbf{s}) &= \sum_{\mathbf{g_1},\cdots,\mathbf{g_N}} p(\mathbf{o},\mathbf{g_1},\dots,\mathbf{g_N}|\mathbf{s}) \\ &= \sum_{\mathbf{g_1},\cdots,\mathbf{g_N}} p(\mathbf{o}|\mathbf{g_1},\dots,\mathbf{g_N})p(\mathbf{g_1},\dots,\mathbf{g_N}|\mathbf{s}) \\ &= \sum_{\mathbf{g_1},\cdots,\mathbf{g_N}} \prod_{i=1}^{N} p(o_i|g_{1i},\dots,g_{Ni})p(\mathbf{g_i}|s_i) \end{aligned} \quad (3)$$

2) Simplified Sensing Model: Because misclassifications have to be taken into account, the general sensing model in (3) is quite expensive to compute. We also provide a simplified sensing model where objects can be missed (unseen) but not misclassified. In such case, each observation random variable O_i becomes dependent directly only on its corresponding state variable S_i, Hence

$$p(\mathbf{o}|\mathbf{s}) = \prod_{i=1}^{N} p(o_i|s_i) \quad (4)$$

Where $p(o_i|s_i)$ expresses the probability of seeing o_i objects of type C_i when there are in fact s_i of them. Under the assumption of independent observations of objects of the same type, the distribution of O_i given $S_i = s_i$ is a binomial $\mathrm{B}(n, p_i)$ with parameters $n = s_i$ and p_i is the probability of seeing an object of class C_i.

B. Deriving the state function $p(\mathbf{s}|r)$

As we consider that each state variable S_j is dependent only on the outcome, the state function $p(\mathbf{s}|r)$ becomes

$$p(\mathbf{s}|r) = \prod_{j=1}^{N} p(s_j|r) \quad (5)$$

Each $p(s_j|r)$ specifies the probability of having exactly s_j objects of type C_j when the outcome of the action is known to be r. To compute $p(s_j|r)$, we use semantic knowledge to derive the implicit expectations $E_r = \{e_1,\dots,e_{n_r}\}$ implied by the outcome. Each expectation expresses a number constraint over the values of the actual number of objects of a certain type C_j, i.e., $e \equiv (S_j \in V_j)$ where $V_j \subseteq \mathbb{N}$.

For instance the action `(move r1 r2)` above has two outcomes. The first one when the robot is still in `r1`, and the second one when the robot moves to `r2`. Since `r1` is a bedroom, and bedrooms are defined in the semantic knowledge base as rooms having at least one bed, at most one sofa, and no sink, the implicit expectations of the first outcome could be $E_1 = \{e_1 \equiv (S_1 \in \{1, 2, 3\}), e_2 \equiv (S_2 \in \{0, 1\}), e_3 \equiv (S_3 \in \{0\})\}$ where S_1, S_2, and S_3 are random variables describing the number of beds, sofas, and sinks respectively that may be in `r1`.

The implicit expectations are then used to compute $p(s_j|r)$ as follows:

- If there is an implicit expectation $e \in E_r$ constraining the values of a state variable S_j, i.e., $e \equiv (S_j \in V_j)$, we should have
$$\begin{array}{lll} p(s_j|r) = 0 & \text{if} & s_j \notin V_j \\ 0 < p(s_j|r) \leq 1 & \text{if} & s_j \in V_j \\ \sum_{s_j \in V_j} p(s_j\,|r) = 1 & & \end{array}$$
The probability mass function $p(s_j|r)$ can be a known mass function used in counting processes such as the binomial or the Poisson mass functions. It can also be specified by the user reflecting her belief.
- For those state variables S_j which are unconstrained in r, we simply assume that $S_j \in \{0, 1, \dots, \max(S_j)\}$,

where $\max(S_j)$ is a predefined upper bound on the values of S_j, and take $p(s_j|r)$ to be a uniform probability mass function.

Example Suppose that we have the movement action `(move r3 r1)` (where `r3` is a living-room and `r1` is a bedroom) with two possible outcomes. In the first outcome, the robot moves effectively inside room `r1`, whereas in the second outcome, the robot stays unintentionally in room `r3`. Deriving the state function for the first outcome implies first deriving the implicit expectations of being in `r1`. Suppose that the concept of `bedroom` as defined in LOOM) gives the following implicit expectations: $e_1 \equiv$ `(at-least 1 has-bed)`, $e_2 \equiv$ `(at-most 1 has-sofa)`, and $e_3 \equiv$ `(exactly 0 has-sink)`.

Then, the conditional probabilities of the state variables given that the robot is in a bedroom, might be determined as follows:

- The number of beds in a bedroom can be modeled as a shifted geometric random variable S_1, i.e.,
$$P(S_1 = i|r) = \lambda(1-\lambda)^{i-1} \quad \text{if} \quad i \geq 1$$
Where λ is the probability of having exactly one bed.
- As sofas are believed to be uncommon in a bedroom, one might have:
$$P(S_2 = 0|r) = 0.8; P(S_2 = 1|r) = 0.2$$
Where S_2 is a random variable modeling the number of sofas.
- The implicit expectation e_3 implies that the random variable S_3 representing the number of sinks has as probability mass function:
$$P(S_3 = 0|r) = 1.0$$
- As there is no implicit expectation about chairs in a bedroom (case 2 above), we can have the uniform probability mass function:
$$P(S_4 = 0|r) = \cdots = P(S_4 = 4|r) = \frac{1}{5}$$
Which means that there might be up to 4 chairs in a bedroom.

Deriving the state function for the second outcome is done in the same way.

Unfortunately there is no workable description logic system which supports probabilistic reasoning (although some attempts have been made in that direction [15]). Therefore, we were obliged to implement the probabilities of the state functions outside the semantic knowledge base.

IV. Information Gathering

There are situations where the available observations are not enough to compute a posterior that matches one of the belief states predicted by the planner. For instance, the planner might predict that after the execution of a "move" action, the robot would be either in a bedroom or the kitchen, but, because the robot did not see a sink, it was not possible to tell with precision where the robot was. One way of coping with this issue is to collect information useful to reach a situation where uncertainty about the outcome of the action is eliminated or at least reduced. To this end, we use information-theoretic measures to select such useful information. One such measure is information gain $IG(R|O_i)$, that expresses the average reduction in the uncertainty – measured by the entropy – in R when the value of O_i is known.

$$IG(R|O_i) = H(R) - \sum_{o_i} p(o_i)H(R|o_i)$$

$H(X)$ is the entropy of the random variable X, that is defined as follows:

$$H(X) = -\sum_{x} p(x)\log(p(x))$$

In our case $IG(R|O_i)$ is computed for all observation random variables O_i in order to select the one that has the highest information gain as the useful information to collect.

V. Test Scenarios

In order to test our approach, we implemented the proposed monitoring framework on a Magellan Pro mobile robot (figure 2) running a fuzzy behavior control architecture for navigation purposes. The robot is equipped with 16 sonars, and a color CCD pan-tilt camera used by the vision system to recognize and classify objects.

Our test scenarios consisted in performing navigation tasks in a house environment. Since our main objective is to show the capacity of monitoring under uncertainty using semantic knowledge 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 house comprises 4 rooms named `r1` to `r4` asserted in the semantic knowledge base as: `r1` and `r2` are of type bedroom, `r3` of type living-room, and `r4` of type kitchen (see figure 2). The semantic knowledge base contains among other things the following concept definitions:

```
(defconcept Room)(defconcept Bed )(defconcept Oven)
(defconcept Sink)(defconcept Sofa)(defconcept TV)

(defconcept Bedroom :is
            (:and Room (:at-least 1 Has-Bed)
                  (:exactly  0 Has-Sink)))
(defconcept Kitchen :is
            (:and Room (:exactly 1 Has-Sink)
                  (:at-least 1 Has-Oven)(:exactly 0 Has-Bed)
                  (:exactly 0 Has-Sofa)))
(defconcept Living-Room :is
            (:and Room (:at-least 1 Has-Sofa)
                 (:exactly 1 Has-TV)(:exactly 0 Has-Sink)))

(tell (Bedroom r1))    (tell (Bedroom r2))
(tell (Living-room r3))(tell (Kitchen r4))
```

We only considered the simple sensing model given in (4), therefore the posterior probability distribution over the outcomes is given by (from equations (4) and (2)):

$$p(r|\mathbf{o}) = \alpha \left[\prod_{i=1}^{N} \sum_{s_i} p(o_i|s_i)p(s_i|r) \right] p(r) \qquad (6)$$

Where the probability parameters of the binomials are all fixed to 0.8. Table I shows the state variables and their domains and table II shows the probabilities of the state functions for S_1, S_2, and S_4 (the probabilities of the state functions for S_3 and S_5 are either 0, 1 or uniform depending on the type of the room) that were used in the test scenarios.

TABLE I
TEST SCENARIOS PARAMETERS

	observable objects (atomic concepts)				
	bed	sofa	sink	oven	TV
i	1	2	3	4	5
S_i values	$\{0,1,2,3\}$	$\{0,1,2\}$	$\{0,1\}$	$\{0,1,2\}$	$\{0,1\}$

TABLE II
STATE RANDOM VARIABLES PROBABILITIES CONDITIONED ON THE TYPE OF ROOMS

	S_1				S_2			S_4		
	0	1	2	3	0	1	2	0	1	2
bedroom	.0	.5	.4	.1	1/3			1/3		
living-room	1/4				.0	.7	.3	1/3		
kitchen	1	0			1	0		.0	.8	.2

A. Negative Evidence

In this test scenario we show how acquiring negative evidence changes the prior probability of the outcomes. We consider two cases: (1) evidence against one outcome, and (2) evidence against all the predicted outcomes of the action.

Case 1. In this test case the robot executed the action `(move r3 r4)` and predicted according to the action model to be in room `r4` (represented as `(robot-in = r4)`) with probability 0.8 and to be in room `r3` (represented as `(robot-in = r3)`) with probability 0.2, i.e.,

$$(R=1) \equiv \texttt{(robot-in = r4)}; P(R=1) = 0.8$$
$$(R=2) \equiv \texttt{(robot-in = r3)}; P(R=2) = 0.2$$

After executing the action, the robot could see only one sink from its current place. The monitoring process was then called to compute the posterior of the outcomes according to the formula in (6), which resulted in $P(R=1) = 1.0$ and $P(R=2) = 0.0$. This result is supported by the fact that room `r3` is a living-room and according to the semantic knowledge it should contain no sink. Therefore seeing a sink is negative evidence against the second outcome.

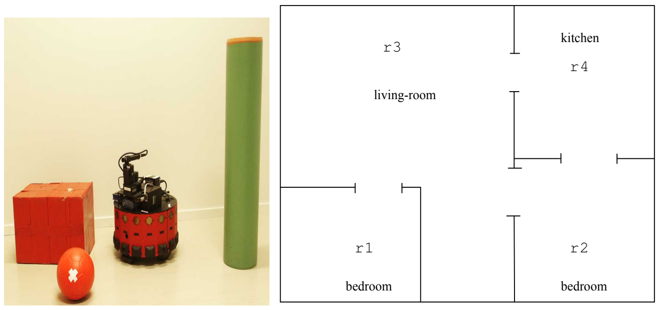


Fig. 2. Experimental setup. (Left) Our robot with simple objects used to represent furniture items. (Right) map of the environment used in our experiments.

Case 2. The aim of this test scenario is to show that the observations acquired by the robot might lead to an exception that it is outside of the action model. In this test case, the robot executed the action `(move r3 r1)` with the outcomes and their probabilities given as:

$$(R=1) \equiv \texttt{(robot-in = r1)}; P(R=1) = 0.7$$
$$(R=2) \equiv \texttt{(robot-in = r3)}; P(R=2) = 0.3$$

As in the previous case, while executing the action, the robot saw only one sink from its place. Computing the posterior resulted in $P(R=1) = 0$ and $P(R=2) = 0$. This result is supported by the fact that room `r1` is bedroom and room `r3` is a living-room and according to the semantic knowledge both should contain no sink. Therefore seeing a sink is negative evidence against both outcomes. This is an exceptional situation, and indicates that something outside of the action model has occurred. In fact, before the robot started the execution of the action, we changed its location and direction so that instead of entering `r1`, it entered `r4`.

To recover from this exceptional situation, our fall-back strategy was to assume that the robot is in one of the remaining locations, i.e., either room `r2` or room `r4` with equal probability $1/2$ and recompute the posterior as before. This yielded $P(\texttt{(robot-in = r2)}) = 0$ and $P(\texttt{(robot-in = r4)}) = 1$, i.e., the robot is certainly in the kitchen.

B. Positive Evidence

In this test scenario, the robot executed the plan action `(move r2 r1)`. As room `r1` is connected to room `r2` through the living-room `r3`, the action could result in being in any one of them. The three possible outcomes of the action and their prior probabilities were:

$$(R=1) \equiv \texttt{(robot-in = r1)}; P(R=1) = 0.5$$
$$(R=2) \equiv \texttt{(robot-in = r3)}; P(R=2) = 0.3$$
$$(R=3) \equiv \texttt{(robot-in = r2)}; P(R=3) = 0.2$$

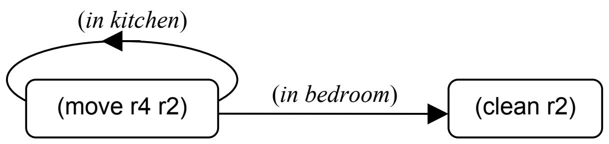
With the robot observing one sofa in its final location, the computed posterior was:

$$P(R=1) = 0.43; P(R=2) = 0.4; P(R=3) = 0.17$$

We can notice that the belief in outcome 2, i.e., `(robot-in = r3)` has increased, while the belief in the other two outcomes has decreased which means that the robot might be in either bedroom despite seeing a sofa, which is a positive evidence for being in a living-room. This is due to the fact that there is no implicit expectation, in the semantic knowledge base, that constrains the number of sofas to be zero in bedrooms. Therefore, seeing a sofa is not negative evidence against being in a bedroom.

C. Information Gathering

We also ran test scenarios related to situations that needed information gathering as discussed in section IV. A typical scenario where information gathering was needed was when the top-level planner produced the following conditional plan for the robot to clean room `r2` starting from room `r4`.



In this test scenario, the model of the action `(move r4 r2)` predicted two outcomes. In the first outcome ($R = 1$) the robot would be in its destination room`r2` with probability 0.73, while in the second outcome ($R = 1$) the robot would still be in room r4 with probability 0.27. Executing the first action `(move r2 r4)` resulted in making no observations. Consequently, the computed posterior of the action outcomes was[1]:

$$P(R = 1) = 0.14; P(R = 2) = 0.86$$

Facing this situation of uncertainty about the location of the robot, the plan executor could not continue the execution of the plan. To be able to do so, the plan executor needed to know whether the robot is either in a kitchen (`r4`) or in a bedroom (`r2`). Therefore, the monitoring process computed the information gain to determine what objects to look for in order to eliminate the uncertainty about the location of the robot. This resulted in selecting objects of type bed to look for as on average observing beds was predicted to achieve the highest information gain of 0.411. Therefore, the robot scanned the room from its current place looking for beds.

We had three runs of this scenario, which all started in the same way (i.e., by executing the action and observing nothing), but continued differently in the information gathering phase. In the information gathering phase of the first run, the robot could see a bed, which resulted in updating the posterior of the outcomes from $P(R = 1) = 0.14$ and $P(R = 2) = 0.86$ to $P(R = 1) = 1.0$ and $P(R = 2) = 0$. Therefore the next action to execute was `(clean r2)`.

In the information gathering phase of the the second run, the robot saw a sink, which resulted in updating the posterior of the outcomes from $P(R = 1) = 0.14$ and $P(R = 2) = 0.86$ to $P(R = 1) = 0.0$ and $P(R = 2) = 1.0$. That meant that the robot was still in the kitchen, thus the next action to execute was `(move r4 r2)`.

Finally, in the information gathering phase of the third run, the robot did not see anything. Although, that changed the posterior probabilities of the outcomes, it did not eliminate the uncertainty. That meant that the plan executor could not continue the execution of the plan. As a result, a failure was raised, and the planner was called to find another plan to achieve the initial task, i.e., clean room `r2`, but this time starting from an uncertain location.

VI. Conclusions

This paper presented a new intelligent plan-execution monitoring approach for mobile robots. The approach is well suited for domains where uncertainty is a key feature, and where semantic domain knowledge is available. We showed how semantic knowledge can be employed to derive expectations, related to action execution, that are used together with action and sensing models to perform execution monitoring.

Our work has been implemented onboard and tested using an indoor mobile robot. As the test scenarios 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 for practical reasons. Our approach could be used with other types of actions (e.g., manipulation, and sensing) whose effects are not related to localization.

There are open issues that we plan to address in our future work. First, a deeper experimental validation is among our objectives. Second, the issue of scalability of the proposed approach is to be investigated carefully. This includes, considering more complex environments as well as dealing with the computational complexity of computing the posterior of action outcomes. We are considering the use of approximate inference methods to tackle the issue of computational complexity.

[1]Notice that seeing no objects increases the belief of being in the kitchen and decreases the belief in being in a bedroom, because the expectations related to a kitchen impose a smaller number of objects to be seen.

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] G. DeGiacomo, R. Reiter, and M. Soutchanski, "Execution monitoring of high-level robot programs." in *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1998, pp. 453–465.
[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] O. Pettersson, "Execution monitoring in robotics: A survey." *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.
[5] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Semantic knowledge-based execution monitoring for mobile robots," in *In Proc of 2007 IEEE Int. Conf. on Robotics and Automation*, 2007.
[6] 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.
[7] J. L. Fernández, R. Sanz, and A. R. Diéguez, "Probabilistic models for monitoring and fault diagnosis: Application and evaluation in a mobile robot." *Applied Artificial Intelligence*, vol. 18, no. 1, pp. 43–67, 2004.
[8] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *Int. Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.
[9] M. Fichtner, A. Großmann, and M. Thielscher, "Intelligent execution monitoring in dynamic environments," in *Proc. of the 18th Int. Conf. on AI, Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments*, 2003.
[10] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An architecture for real-time reasoning and system control," *IEEE Expert*, vol. 7, no. 6, pp. 34–44, 1992.
[11] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook*. Cambridge University Press, 2003.
[12] R. MacGregor, "Retrospective on loom," Information Sciences Institute, University of Southern California, Tech. Rep., 1999.
[13] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space." in *Proc. of the 5th Int. Conf. on AI Planning Systems*, 2000, pp. 52–61.
[14] L. Karlsson, "Conditional progressive planning under uncertainty," in *Proc. of the 17th Int. Joint Conf. on AI*, 2001, pp. 431–438.
[15] D. Koller, A. Y. Levy, and A. Pfeffer, "P-classic: A tractable probabilistic description logic," in *Proc. of the AAAI*, 1997, pp. 390–397.