Monitoring The Execution of Robot Plans Using Semantic Knowledge

Abdelbaki Bouguerra and Lars Karlsson and Alessandro Saffiotti

AASS Mobile Robotics Lab, Örebro University, Örebro, Sweden

Abstract

Even the best laid plans can fail, and robot plans executed in real world domains tend to do so often. The ability of a robot to reliably monitor the execution of plans and detect failures is essential to its performance and its autonomy. In this paper, we propose a technique to increase the reliability of monitoring symbolic robot plans. We use semantic domain knowledge to derive *implicit* expectations of the execution of actions in the plan, and then match these expectations against observations. We present two realizations of this approach: a crisp one, which assumes deterministic actions and reliable sensing, and uses a standard knowledge representation system (LOOM); and a probabilistic one, which takes into account uncertainty in action effects, in sensing, and in world states. We perform an extensive validation of these realizations through experiments performed both in simulation and on real robots.

Key words: Plan Execution and Monitoring, Mobile Robots, Semantic Knowledge

1 Introduction

Plan-based control architectures have been widely used for the control of mobile robots that are designed to achieve a multitude of complex tasks in real-world environments [1–3]. A major challenge in plan-based control is how to make sure that the actions in the plan are executed correctly and reliably. Consequently, plan execution monitoring aims at detecting situations where the actual outcome of an action diverts from the intended one. Model-based approaches to plan execution monitoring are based on the idea of comparing the *explicit effects* of each action (that is, the ones stated in the model of that action) to what is observed after the execution of that action [4]. This

Email address: aba;lkn;asaffio@aass.oru.se (Abdelbaki Bouguerra and Lars Karlsson and Alessandro Saffiotti).

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

supposedly means that the effects to monitor are directly observable. For example, a mobile robot that has executed the planned action `(enter r1 d1)`, to enter the living-room `r1` through door `d1`, would query its self-localization system to verify that the explicit expectation `(robot-in r1)` is indeed the case. This way, execution monitoring completely relies on the accuracy of the self-localization system. In this article, we propose to increase the reliability of execution monitoring by incorporating more advanced forms of reasoning. In particular, we propose to use semantic knowledge about the domain to derive *implicit expectations* about the effects of actions, and to monitor these expectations. By implicit expectations we mean expectations that can be logically derived from the explicit ones through the use of semantic knowledge [1]. In the above example, since `r1` is an instance of the class `Living-Room`, the robot should expect to see objects that are typical of a living-room, such as a TV-set and a sofa. If the robot sees an oven, it should conclude that it is not in the living-room, and henceforth that the execution of `(enter r1 d1)` has failed. Another example is grasping a coffee cup: semantic knowledge can be used to generate and check the implicit expectation that the object in the gripper has properties such as being a container and having exactly one handle.

In this paper, we define the notion of *Semantic Knowledge-based Execution Monitoring* (*SKEMon*), and we propose a general algorithm for it based on the use of description logics for representing knowledge. We also develop a second process to take into account probabilistic uncertainty both in acting and sensing. In particular, we allow for sensing to be unreliable, for action models to have more than one possible outcome, and we take into consideration uncertainty about the state of the world. This is essential to the applicability of our technique, since uncertainty is a pervasive phenomenon in robotics.

This article builds upon and consolidates our previous work on semantic knowledge-based execution monitoring [5,6]. In addition, this article reports a new extensive experimental evaluation of two processes of our monitoring approach: a crisp (boolean) process, and a probabilistic process. These experiments show that the use of semantic knowledge contributes to more reliable execution monitoring. They also show that the crisp version mainly works by finding counter evidence, while the probabilistic version is able to better take positive evidence into account. The systematic evaluation was performed in simulation, but we also include experiments to demonstrate that our approach can be made operational on a real robot.

The paper is organized as follows. In section 2, we review related work in plan execution monitoring. In sections 3 and 4, we describe our two execution

[1] Implicit expectations could also be taken into account inside the planning process. This, however, would greatly increase the complexity of the planner. In our approach, implicit expectations are computed at execution-time.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

monitoring processes: crisp and probabilistic. We present our experimental validation in section 5 and conclude in section 6.

2 Related Work

To the best of our knowledge, no published research work has used semantic domain-knowledge to derive and monitor implicit expectations related to the correct execution of robot plans. In this section, we review research work that has addressed execution monitoring of mobile robot actions from one point or another. As mentioned above, traditional approaches focus on comparing the model-based and the actual states of the world. In general, predefined models are used to describe the outcomes of an action when executed successfully. Examples of architectures in this category include the PLANEX system [1], the LAAS architecture [2,7], and ROGUE [8]. Using only action models allows only the verification of effects explicitly encoded in those models. In our work, we focus on monitoring conditions that are not explicitly encoded in the models of the planning actions. To achieve this, we use extra general domain-knowledge to derive and very implicit expectations that are consequences of the explicit effects of actions. It should be noted that there are monitoring approaches that do not use explicit models at all. For instance, Pettersson *et al.* [9] propose to let robots learn to recognize patterns of behavior activation that indicate failure.

There are some execution monitoring approaches that use logic formalisms to describe the dynamics of the environment. However, these approaches focus on the explicit effects of actions. An example of a logic-based approach is the work of De Giacomo *et al.* [10] describing a process for monitoring the execution of robot programs written in *Golog*. The working of *Golog* is based on the Situation Calculus, which is a logical formalism for reasoning about the consequences of actions. The execution monitor compares what the robot expects and what it senses to detect discrepancies and recover from them. Discrepancies are assumed to be the result of exogenous actions. The recovery is done through a call to a planner to produce a *Golog* program consisting of a sequence of actions that locally transform the current situation to the one expected by the original program before it failed. The work by Fichtner *et al.* [11] employs the Fluent Calculus, a logical action formalism, to model actions and their effects. Besides detecting discrepancies, the authors describe how such a formalism can be used to provide explanations of why failures occurred, which can be useful to recover from such failures. Lamine and Kabanza propose to use Linear Temporal Logic (LTL) with fuzzy semantics to encode knowledge about successful execution of robot actions [12]. Such knowledge is used by the monitoring process to check the correct execution of the robot actions by considering not only present execution information, but also past one.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Reactive execution systems implementing the BDI model, such as PRS [13], RAPS [14], and the commercial JACK system [39] typically use hand-coded procedures to monitor events that might affect the execution of the agent actions. Consequently, expectations are explicitly coded in the monitoring procedure. Thus handling new events implies writing new monitoring procedures. Regarding plan execution monitoring under uncertainty, we cite the work by Fernández *et al.* [15] where Partially Observable Markov Decision Processes (POMDP) are used to generate action policies that include detecting and recovering from unexpected situations. In a related work by Verma *et al.* [16], Bayesian filtering techniques are employed to detect and diagnose exceptional situations caused by hardware faults in planetary rovers. In comparison, we handle uncertainty at execution time by taking into account only the different possible outcomes of the executed action.

There are also other approaches that monitor conditions other than the explict effects of actions. For instance, Fraser *et al.* [17] describe an approach that considers monitoring plan invariants, i.e., environment conditions that have to hold during the whole execution episode of a plan; Fernández and Simmons [18] use a hierarchy of monitors designed to detect symptoms of specific exceptional situations; Beetz and McDermott [19] propose to debug plans while they are executed to prevent probable execution failures. The Rationale-Based Monitoring approach [20] monitors features of the environment that can affect the plan under construction.

The work by Galindo *et al.* presented in [21] 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.

3 Semantic Knowledge in Execution Monitoring

Semantic knowledge refers to knowledge about objects, their classes and how they are related to each other (this knowledge is sometimes called "ontological"). For instance, an office is a concept that refers to rooms that have at least one desk and a chair; the entities desks and chairs are themselves defined as pieces of furniture, etc. Semantic knowledge can be used by mobile robots to help them communicate with humans [22], to analyze scenes [23], and to build maps [21,24]. We use semantic knowledge in the process of monitoring the execution of symbolic plans, i.e., actions that use symbols to refer to physical objects. An example of such an action is `(pick-up c)` where `c` is a symbol that refers to a cup. The robot can use knowledge that cups are containers

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

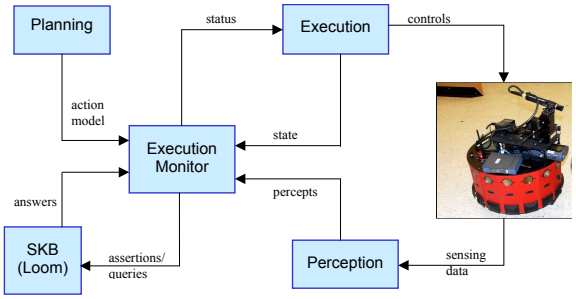

Fig. 1. The different modules involved in monitoring implicit expectations (only the arrows relevant to execution monitoring are shown).

that have one handle to detect incorrect execution situations whenever the grasped object does not have exactly one handle.

Figure 1 shows the different modules involved in monitoring the execution of symbolic plans using semantic knowledge. The execution module keeps track of the current plan and its execution context. It also keeps track of the current estimated state including the current expected location of the robot. It is in charge of translating the high-level actions into executable processes. The perception module provides the monitoring module with perceptual information computed from the raw data coming from the robot's on-board sensors such as camera and laser. The perceptual information is expressed in a symbolic form that describes objects and their observable properties (e.g., color, shape, marks, and relative position). The planning module provides the monitoring module with the model of the action whose execution is to be monitored. This model specifies the preconditions that should hold in the real world for the action to be applicable. The model specifies also the explicit effects of the action that should result when the action is executed. The explicit effects are divided into negative and positive effects. Negative effects express conditions that should not hold, while positive effects encode conditions that should hold when the action is executed successfully. The role of the semantic knowledge base (SKB) is to store general and assertional domain knowledge. It also processes and answers queries related to its contents. Finally, the monitoring module itself is in charge of making sure that actions are executed successfully, by comparing the expected consequences of an action to what has been perceived by the perception module.

3.1 Representing Semantic Knowledge

We opted for description logics (DLs) [25] to represent and reason about semantic knowledge, since they provide a good trade-off between representation power and reasoning tractability. An important characteristic of DLs is their reasoning capabilities of inferring implicit knowledge from the explicitly rep-

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

resented knowledge. In DL formalisms, unary predicates represent concepts (also called classes), i.e., sets of individuals, and binary predicates express relationships between individuals.

We use LOOM [26], a well established knowledge representation and reasoning system, for modeling and managing semantic domain knowledge. LOOM 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. In fact, the work described in this article can be implemented using other DL-based knowledge representation and reasoning systems. LOOM provides a definition language to write definitions of concepts and relations, and an assertion language to assert facts about individual objects. LOOM supports a first order query language to retrieve instances from a knowledge base. It is also possible to ask the knowledge base whether or not a proposition is true according to the conceptual definitions. LOOM uses open world semantics as the default assumption when trying to prove or disprove a proposition. This makes it possible to conclude whether the truth value of a proposition is true, false, or simply unknown.

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 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. More complex concept expressions are constructed by combining other concept names using a limited number of connectives (`and, or, not, implies`). Concepts that are not defined in terms of other concepts are called atomic concepts (see the appendix for examples of more complex definitions of concepts).

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 automatically as an instance of the class `room`. Classi-

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

SKEMon(*obj*)

1. $CLs \longleftarrow$ SKB::get-asserted-classes(*obj*)
2. $temp \longleftarrow$ PERCEPTION::perceived-object
3. $\Pi \longleftarrow$ PERCEPTION::perceived-properties&relations(*temp*)
4. SKB::create-instance $(temp, \Pi)$
5. **if** $\forall\ cl \in Cls.$ SKB::is-instance-of(*temp*, *cl*) **then** *success*
6. **else if** $\exists\ cl \in Cls.$ SKB::is-not-instance-of(*temp*, *cl*) **then** *failure*
7. **else** *unknown outcome*

End

Fig. 2. The pseudo code of the semantic knowledge-based execution monitoring process *SKEMon*.

fication 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 are asserted (added to the knowledge base), they are classified into that taxonomy.

3.2 The Monitoring Process

A process for Semantic Knowledge-based Execution Monitoring which we call *SKEMon* is outlined in figure 2. The process typically checks if an execution-time object fits the description of an expected object. For instance, if the action to execute is `(pick-up c1)` to pick up object `c1` (the expected object), then the object actually picked up by the robot is the execution-time object; and thus needs to be checked to verify if it matches the description of `c1`. The execution of the navigation action `(enter r1)` implies that the execution-time object is the room where the robot ended up, which needs to be checked against the description of `r1` (the expected object). In figure 2, the operations prefixed by "SKB::" involve using the semantic knowledge base, whereas those prefixed by "PERCEPTION::" involve the perception module (see figure 1).

The process gets the name of the expected object `obj` which is derived from the action model; in our current implementation `obj` is derived from the positive effects of the executed action. The process starts by querying LOOM (the SKB) about the asserted classes of the expected object `obj` (step 1). Only the most specific asserted classes are considered, since the semantic knowledge base can deduce that an instance of a specific class is also an instance of all the more general classes. For instance, if `r1` is asserted once to be a room and once to

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

be a living-room, then only the living-room class is considered.

In step 2, the perception module is asked to return the execution-time object which is given a temporary name by the *SKEMon* process. Then, the perception module is queried about the perceived properties and relations (to the other perceived objects) of the execution-time object (step 3). It should be noted that in our current implementation, the perception module retains only percepts that are relevant to the current domain by filtering the stream of percepts coming from the vision system. The filtering is carried out by hard-coded functions that are defined as part of each domain. An alternative solution would be to automatically construct filters using the definitions of concepts and relations which are stored in the semantic knowledge base.

In step 4, the perceptual information about the execution-time object is used to create a temporary instance in SKB. 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 asserts those facts in the SKB by issuing the following LOOM command:

```
(tell (room temp)(has-chair temp ch1)(has-bed temp b1))
```

where `temp` is a temporary symbol used to refer to the current room (where the robot is actually located), i.e., the execution-time object. LOOM classifies the newly created instance based on the properties and relations to the other perceived objects, i.e., the chair `ch1` and the bed `b1`. Once LOOM is done with the classification of the execution-time object, the monitoring process sends another query to LOOM to check whether the classification is consistent with the asserted classes of the expected object *obj* (step 5). In step 6, the monitoring process checks whether the available perceptual information reveals that one of the constraints, involved in the definition of the classes of the expected object, is violated. For our example, this is performed by sending the following two queries to LOOM where the second query is asked only when the answer to the first one is "*NO*".:

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

The monitoring process interprets LOOM's answers as follows:

- **Consistent Classification.** A *YES* on the first query means that the implicit expectations are verified and the execution-time object is classified like `obj`. Therefore, the *SKEMon* process reports *success* (step 5). In our example, this means that the robot is in the right type of room.
- **Inconsistent Classification.** A *YES* on the second query means that the

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

	m < n	m = n	m > n
`(:at-least n R)`	unknown	YES	YES
`(:exactly n R)`	unknown	unknown	NO
`(:at-most n R)`	unknown	unknown	NO

Table 1
Truth values of number constraints given in function of `m`: the number of objects that have been observed to be related by relation `R` to a specific individual.

classification of the execution-time object is inconsistent with the expected object `obj`. Hence, *failure* is reported (step 6). In our example, this means that the robot is dislocated.

- **Unknown Outcome.** *NO* on both queries means that it cannot be determined whether some constraints (implicit expectations) hold or not (step 7). In our example, if no sofa is observed and 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.

The *unknown* outcome is due to the fact that we set up LOOM to operate according to open world semantics. In other words, the facts told to LOOM are assumed to be only a part of the complete world state. As a result, LOOM assumes that there might be additional facts of which it has not been told. The reason behind using open world semantics is to be able to take into account partial observability of the environment: due to occlusions, the robot gets only partial information about the presence of objects and their properties. Using open world semantics makes number constraints prone to give an *unknown*. An `(at-least n R)` constraint gives *unknown* whenever the total number of observed objects related to the constraint is less than the lower bound `n`. The constraint gives *YES* otherwise. An `(at-most n R)` constraint gives *unknown* as long as the total number of observed objects related to the constraint is not above the upper bound `n`. The constraint gives *NO* otherwise. Table 1 shows the answers to queries about the truth value of three number constraints given as a function of the total number `m` of observed objects.

There are two options to handle the *unknown* outcome. The first option is to be credulous and consider the absence of counter-evidence as sufficient grounds for assuming that the execution of the action has succeeded. In our example, this implies that the monitoring process should 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, the monitoring process assumes that the location is correct as long as no evidence of the contrary is detected. The second option is to take a cautious approach and actively try to gather more information in order to do a better classification. The reader is referred to [27] for more information on how sensor-based planning can be used to

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

collect information for the purpose of monitoring using semantic knowledge.

3.3 Dealing with Unexpected Situations

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. The recovery procedure consists in finding a sequence of actions that would lead to a situation where the robot can continue executing its top-level task plan. In our navigation example, replanning is needed when the robot is found to be dislocated. The first step in replanning is the creation of a world state that reflects the resulting unexpected situation, i.e., update the location of the robot 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 the recovery module should take this fact into account.

4 Handling Uncertainty

The *SKEMon* process presented in the previous section is limited to action models with deterministic effects. The process has also the limitation of treating expectations in a boolean fashion, i.e., evaluated to be either true, false, or unknown. These limitations are mainly due to the fact that the process does not represent uncertainty inherent in action effects, world states, and sensing. In this section, we describe a different execution-monitoring process that is able to reason about uncertainty. To this end, we develop a model that takes into account quantitative uncertainty in the form of probabilities in states, actions, sensing and the way semantic knowledge is used to interpret expectations. More specifically, action models can encode different outcomes each with a given probability of occurrence, and sensing can be noisy. As a result of using probabilities, it is possible to go beyond a boolean treatment of whether an expectation is verified. In other words, the monitoring process can compute a probability for whether a certain expectation is verified, such as "the robot is in the kitchen with 0.85 probability" instead of returning "unknown" as a monitoring result. Moreover, the fact that the a posteriori probability of each outcome of an action can be estimated enables a more informed decision about how to proceed (consider action execution successful, failed, or more information needed) than with just a boolean approach. Typically, the probabilistic monitoring process works as follows:

- For each possible outcome of the action whose execution is being monitored,

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

a set of implicit expectations are determined.
- Those expectations are used to estimate a probability distribution over actual world state. For instance, the implicit expectation of seeing at least one desk implies that the probability of having no desk is zero, while the probability of having one, two, or more desks is strictly greater than zero. Although reasoning under uncertainty in description logics is an ongoing research activity [40], currently there is no available DL system that supports probabilities. Thus, the probability distributions of the expected state of the world are computed by a precoded procedure.

Besides uncertainty about the world state, uncertainty in sensing is taken into account by 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, e.g., to take occlusions into account.
- How a seen object is classified, i.e., the model accounts for misclassification of objects when they are seen. For instance, a sofa may sometimes be mistaken to be an arm-chair.

The monitoring process 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 [28]. 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:

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

where $p(\mathbf{o})$ is a normalizing factor. To compute the posterior $p(r|\mathbf{o})$, two probability functions are needed: (1) the prior probability $p(r)$ which is derived from the action model, and (2) the observation function $p(\mathbf{o}|r)$. In the following we show how the latter is computed. We adopt the following notation: bold-face letters denote vectors such that the i^{th} element of a vector $\mathbf{X}$ is denoted by X_i. Capitalized letters denote variables, while uncapitalized letters denote specific values, e.g., o is the same as $O = o$ and $\mathbf{x}$ is the same as $\mathbf{X} = \mathbf{x}$.

We restrict ourselves to constraints over atomic classes of observable objects. One could easily add constraints over values of other attributes, e.g., `color` $\in \{$ `red, yellow, white`$\}$. In our model we use the following entities:

- R: a random variable denoting the different outcomes of the executed action.
- $\mathbf{O}$: a random vector such that its ith random variable O_i represents the

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

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.

- **S**: a random vector such that its ith component is a state variable whose values are the actual number of objects of type C_i. Each state variable S_i takes values in a finite domain $V_i \subset \mathbb{N}$. In our model, each S_i depends directly only on R. The size N of **O** and **S** is the number of the observable atomic concepts C_i.

Consequently, equation (1) becomes:

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

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 $p(\mathbf{o}|r)$, 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 $\mathbf{s}$, and (2) a state function $p(\mathbf{s}|r)$ that describes the probability of $\mathbf{s}$ when the outcome of the action is r.

Example. Consider the execution of the navigation action `(move loc1 loc2)` whose model accounts for two possible outcomes. The first outcome, i.e., $R = 1$, is when the robot remains unintentionally in `loc1`, while the second outcome, i.e., $R = 2$, is when the robot moves effectively to `loc2`. If the only classes of observable objects that can exist in either location are beds and sinks, then S_1 and S_2 denote respectively the actual number of beds and sinks that exist in one of `loc1` or `loc2`. O_1 and O_2 denote respectively the number of observed beds and sinks in the current location.

4.1 The Sensing Model $p(\mathbf{o}|\mathbf{s})$

The function $p(\mathbf{o}|\mathbf{s})$ specifies the probability that the robot will observe o_1 objects of class C_1, o_2 objects of class C_2, $\cdots$, given that the actual values of the state variables are equal to $\mathbf{s}$. In its general form $p(\mathbf{o}|\mathbf{s})$ permits to specify if an object is seen or not, and how a seen object is classified, i.e., the model accounts for misclassification of objects when they are seen.

The potential for misclassifying objects when they are seen implies that all random variables in **O** and **S** depend on each other. As a result, there is an exponential number of probabilities $P(\mathbf{O} = \mathbf{o}|\mathbf{S} = \mathbf{s})$ 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 $g_{ik}(k = 1 \ldots N)$ objects of class C_k. The number of missed (unseen) objects of type C_i is denoted by $g_{i(N+1)}$.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

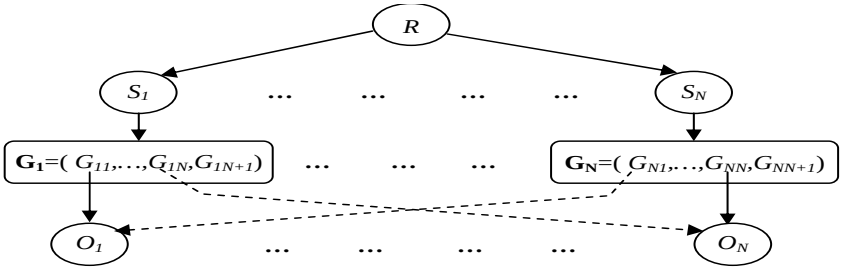

Fig. 3. The dependency structure of the different random variables used in the state and sensing functions.

Under the assumption of independent classification of objects of the same class, $p(\mathbf{g}_i|s_i)$ can be expressed by a multinomial probability mass function. The parameters n and $p_{k(1\leq k\leq N+1)}$ of the multinomial are defined as $n = s_i$ and p_k is the probability of classifying an object of class C_i as of class C_k (for $k \leq N+1$) while p_{N+1} is the probability of missing (not seeing) an object of class C_i. Figure 3 shows the dependency structure of $R, \mathbf{S}$, $\mathbf{O}$, and $\mathbf{G_i}$. Note that o_i represents the total number of objects classified as C_i; either correctly or incorrectly (e.g., the number of observed chairs is the total number of objects classified correctly or incorrectly as chairs), thus we have $p(o_i|g_{1i}, \ldots, g_{Ni}) = 1$ when $\sum_{k=1,N} g_{ki} = o_i$. Finally, the sensing model is formulated as:

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

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

The state function is where semantic knowledge is encoded. This function gives, for instance, the probability that the grasped object has a handle given that it is a cup, or the probability that a room has a stove given that it is a kitchen. Unfortunately there is no workable description logic system which supports probabilistic reasoning (although some attempts have been made in that direction [29]). Therefore, we were obliged to implement the probabilities of the state functions outside the semantic knowledge base.

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

$$p(\mathbf{s}|r) = \prod_{j=1}^{N} p(s_j|r) \tag{4}$$

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Each $p(s_j|r)$ specifies the probability of having exactly s_j instances of class C_j given that the outcome of the action is known to be r. To compute $p(s_j|r)$, we use semantic knowledge to detrmine the implicit expectations $E_r = \{e_1, \ldots, e_{n_r}\}$ implied by the outcome r. Each expectation expresses a number constraint over the values of the actual number of objects of a certain class C_j, i.e., $e_j \equiv (S_j \in V_j)$ where $V_j \subseteq \mathbb{N}$. The implicit expectations are then used to compute $p(s_j|r)$ as follows:

- For expectations constraining the values of a state variable S_j, i.e., $e_j \equiv (S_j \in V_j)$, we should have: $0 < p(s_j|r) \leq 1$ if $s_j \in V_j$ otherwise $p(s_j|r) = 0$; we also should have $\sum_{s_j \in V_j} p(s_j\ |r) = 1$.
  The probability mass function $p(s_j|r)$ can be chosen to be a known function used in counting processes such as the binomial or the Poisson mass functions. It can also be given as a table of probabilities reflecting the belief of the user. Using a known probability mass function reduces considerably the amount of information that the user has to provide, since only a few parameters need to be specified.
- For state variables S_j that are unconstrained in r, we simply assume that $p(s_j|r)$ is a uniform probability mass function.

Example. Continuing our example of `(move loc1 loc2)`, the implicit expectations of being in `loc1` are determined based on the type of `loc1`. If `loc1` is asserted to be a bedroom and bedrooms are defined as rooms having at least one bed and no sink, then the implicit expectations could be $E_1 = \{e_1 \equiv (S_1 \in \{1, 2, 3\}), e_2 \equiv (S_2 \in \{0\})\}$. The conditional probabilities of the state variables given that the robot is in a bedroom might be determined as follows: the number of beds, i.e., S_1 in a bedroom can be modeled as a shifted geometric function, 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 in a bedroom. The implicit expectation e_2 implies that $P(S_2 = 0|R = 1) = 1.0$.

5 Experiments

We have implemented the two monitoring processes on a real robot and ran tests in real world to demonstrate the feasibility of the approach. We also conducted experiments in simulation to collect large amounts of data for the purpose of statistically evaluating the performance of the proposed framework. Due to lack of benchmark systems in execution monitoring of symbolic plans, we base our evaluation on the metrics of false positive rate (FPR, the ratio between the number of false positives and the total number of actual negative cases) and true positive rate (TPR, the ratio between the number of true positives and the total number of actual positive cases).

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

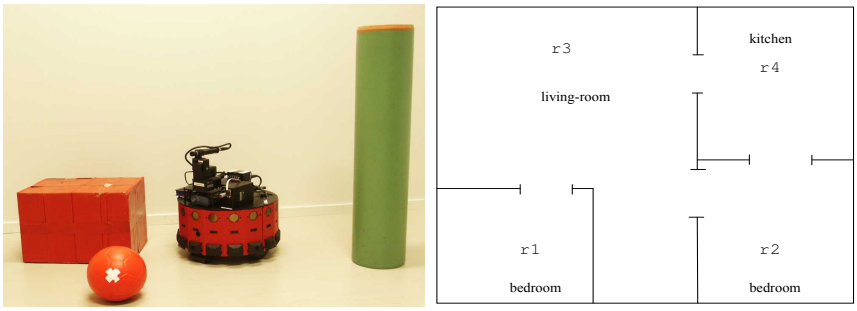

Fig. 4. Experimental setup. (Left) Pippi, the Magellan Pro robot together with simple objects used to represent furniture items. (Right) Map of the environment used in performing test cases.

5.1 Real Robot Test Scenarios

In order to test the *SKEMon* process, we implemented both versions, i.e., crisp *SKEMon* and probabilistic *SKEMon*, on a Magellan Pro mobile robot, called Pippi, (figure 4) running a fuzzy behavior control architecture for navigation purposes [30]. The robot is equipped with 16 sonars, and a color CCD pan-tilt camera used by the vision system to recognize and classify objects. Since our main objective is to show the capacity of execution monitoring using semantic knowledge and not object recognition, we use a simple vision system that is able to recognize objects with simple shapes using color segmentation. It should be noted that our approaches do not depend on the simple vision system we used. One can use more robust object recognition and classification systems such as the system described in [31] that uses scale and orientation invariant local descriptors (SIFT features) [32] to identify objects occurring in typical household environments.

Our test scenarios consisted only in performing navigation tasks in a house environment (see figure 4) where we let simple shapes like cylinders and boxes stand in for beds, sofas, etc. The test runs reported below have been performed in a lab environment, placing the simple objects to simulate pieces of furniture. The semantic knowledge base is given in appendix A.

5.1.1 Crisp SKEMon Test Cases

We start by describing test cases where crisp *SKEMon* using a credulous approach was used to check the implicit expectations of navigation actions. Pippi was assigned tasks to clean the different rooms in the house. Plans for achieving those tasks were all generated under the assumption that the actions would be reliably executed and that there was no sensing noise. In the following, we describe test cases of monitoring instances of the `(enter ?loc)` action.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Correct Success Result 1. In this test case, Pippi finished the execution of `(enter r4)` and could see only an oven from its final place. After asserting the fact that the current room has one oven and since ovens are defined to be exclusively in kitchens, LOOM classified the current room as an instance a the class `kitchen`. As a result, the *SKEMon* process returned *success* (step 5 of the process) without even having to consider a credulous approach. This test case shows that not all the implicit expectations need to be verified in order to conclude that an action has been successfully executed. It was enough to see objects that are characteristic of a certain class of rooms to deduce the class of the room where the robot was.

Correct Success Result 2. The aim of this test case is to show that the monitoring process uses the absence of counter evidence to correctly conclude that an action is executed successfully, although some implicit expectations are not known to hold or not. Here, Pippi started in `r4` and executed the action `(enter r3)` to enter the living-room `r3`. Pippi could perceive one sofa from her final position. That did not help LOOM to know whether the current room was an instance of the class `living-room`, i.e., *SKEMon* reached an *unknown* result (step 7 of the process shown in figure 2). Nevertheless, Pippi was concluded to be in `r3`, since the *SKEMon* process was using a credulous approach. Notice that the *unknown* result was returned because sofas were not defined to be exclusively constrained to be in living-rooms.

False Success Result. This test case shows that the credulous approach might cause the monitoring process to wrongly conclude that the execution of an action has succeeded. Here, Pippi finished the execution of the action `(enter r1)` but instead of entering bedroom `r1` she ended up in the kitchen where she could see only a table. As tables could be in either room, the *SKEMon* result was *unknown*. Using the credulous approach suggested that Pippi was in `r1` which was a false positive.

Correct Failure Result. The aim of this test case is to show that it is enough that one implicit expectation is violated to conclude that the execution of the action has failed. We repeated the previous test case, but instead of seeing a table, Pippi saw a sink. This meant that the *SKEMon* process could conclude that the execution of the action has failed, since the implicit expectation of having no sink in a bedroom was violated.

5.1.2 Probabilistic SKEMon Test Cases

We also run test cases where uncertainty in action effects and sensing were reasoned about by the probabilistic version of *SKEMon*. The parameters used in the sensing model and the state functions are given in appendix B. We considered monitoring the execution of the action `(move ?loc1 ?loc2)` with a

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

model of a prior giving 20% chance for the robot being stuck in room `?loc1` (outcome 1, denoted $R = 1$) and 80% chance for the robot ending up in room `?loc2` (outcome 2, denoted $R = 2$). Thus, the *SKEMon* process returned the outcome that had the highest posterior probability.

Negative Evidence Against One Outcome. In this test case we show how acquiring negative evidence changes the prior probability of the outcomes. Here, Pippi started in room `r3` and executed the action `(move r3 r4)` to move to room `r4`. Pippi effectively moved into `r4` and could perceive only one sink from its final place. The computed posterior was $P(R = 1) = 0.0$ and $P(R = 2) = 1.0$. This result was supported by the fact that room `r3` was a living-room and according to the semantic knowledge it should contain no sink. Moreover, in the sensing model, the only object that could be mistaken as a sink was an oven which should not be found in a living-room either. Therefore seeing a sink was negative evidence against the first outcome.

Uncertain Posterior. We considered two test cases where perceptual information did not eliminate the uncertainty about action outcomes. Both test cases involved monitoring the execution of the same action `(move r3 r1)`, where Pippi started from the same location in room `r3` and effectively moved into room `r1`. In the first test case, Pippi could see 2 chairs from its final place. As chairs could be in both rooms, the probabilistic *SKEMon* process reached the posterior $P(R = 1) = 0.13$ and $P(R = 2) = 0.87$. This was a predictable result as the conditional probability of seeing chairs in both rooms was the same. In the second case, Pippi could see only a sofa. The computed posterior was $P(R = 1) = 0.4$ and $P(R = 2) = 0.6$. This result was due to the fact that in the sensing model, seeing a sofa was interpreted as either a sofa or a bed, with different probabilities of course.

5.2 Simulation Results

We used the 3D robot software simulator "Gazebo" [33] to run our simulation experiments. We collected data of monitoring the execution of manipulation and navigation actions inside a house environment. A mobile robot called Astrid, of type ActiveMedia PeopleBot, was used in both scenarios as the main protagonist (fig. 5). The robot is equipped with a color camera that was used to acquire visual perceptual information about the environment.

5.2.1 Manipulation Scenario

In the first scenario, we used a simulation of the smart house described in [34]. Besides the robot, the experimental set-up includes a two prismatic-joint arm that is attached to the roof of a fridge (see figure 5). The arm is used

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

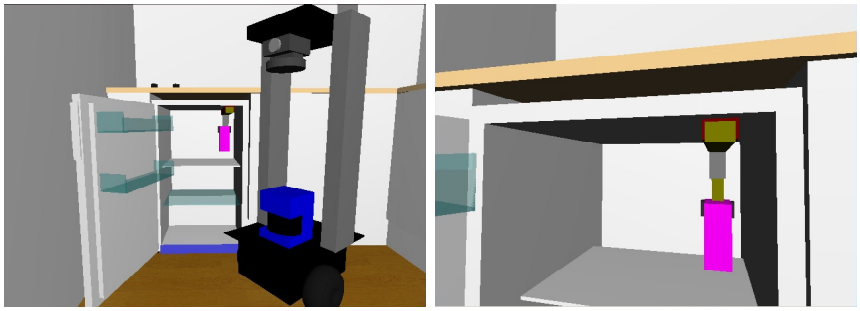

Fig. 5. Simulation experimental setup. (Left) Astrid the robot and the fridge. (Right) A close up of the arm while picking up an object inside the fridge.

for picking up objects inside the fridge and place them on the base of the robot so they can be carried to another location inside the house. All objects belong to subclasses of a general class `container`, and each subclass had some specific properties in terms of constraints over relations to other primitive objects (handle, cap, and cover). The semantic definitions of such classes are relatively simple, and they are given in appendix A.

We used the two simple solids of a cylinder and a box to represent a container, while the related objects (i.e., handle, cover, cap) were represented by marks of different colors placed on the primary objects. For example, an object of type cup was represented by a box (container) that had a yellow mark (handle).

5.2.2 Navigation Scenario

In this scenario, Astrid is acting in a house environment that comprises rooms of different types (bedroom, living-room, kitchen, bathroom, utility room, and office). In each room there are furniture items that are typical of the type of that room. For instance, in a kitchen, there's at least one oven, at least one sink, etc. In total, there are thirteen different types of objects that can exist in a room. The semantic definitions of the different types of rooms and furniture items is given in appendix A. The semantic knowledge used in this scenario is more complex than in the previous scenario. The definitions contain more constraints, and there are more related objects to take into account in order to classify a room. Moreover, there are certain types of objects that, when observed, do not contribute to the classification process, like plants. As in the manipulation scenario, the furniture items were represented by objects of simple shapes and colors.

5.2.3 Perceiving the Environment

We vary the degree of partial observability of the environment using a parameter P_{perc} that specifies the probability of perceiving all objects related to the

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

actual outcome of an executed action. In our experiments, we assume that all the objects have the same probability of being perceived, i.e., $\sqrt[m]{P_{perc}}$, where m is the total number of objects that are related to the actual outcome of the action. We also assume that these probabilities are independent.

5.2.4 Crisp SKEMon

We tested the performance of the crisp *SKEMon* process on both scenarios. For the manipulation scenario, each experiment consisted of executing the high-level action "`(pick-up obj)`" by the arm to pick up the object "`obj`" that was inside the fridge. The high-level definition of the pick-up action is given as follows:

```
(ptl-action :name (pick-up ?obj)
 prec:    (and (arm-empty)(inside-fridge ?obj))
 results: (and (arm-empty = f)(holding = ?obj)))
```

The *SKEMon* process was called once the low-level execution process reported that it succeeded in picking up the desired object. The camera on-board the robot was used for acquiring perceptual information about the picked up object. That meant the robot had to be in front of the fridge facing one side of the picked up object. Thus, the related object (handle, cap, or cover) could be observed only if it was on the side facing the robot.

The "`(pick-up obj)`" experiment was run such that `obj` was asserted 100 times to be one of each of the five types: cup, bowl, bottle, glass, and box, giving a total of 500 runs. In each run the type of the object that was actually picked up was uniformly sampled from the five available types. Whether the related object (e.g., handle, cap,...) was visible was decided by using a Bernoulli distribution with success probability P_{perc}.

Similarly, we conducted a total of 600 runs of the `(enter loc)` navigation action to enter a room identified by the symbol `loc` and whose type was asserted to be one of the available room types, i.e., bedroom, living-room, etc. Each room type was considered 100 times. Each time, the type of the actual final location of the robot was sampled uniformly from the six available types. A world state, containing objects that were consistent with the actual location, was then generated using the state functions used in the probabilistic version of *SKEMon* (see appendix B). The objects that could be perceived from the robot's position were determined according to the P_{perc} parameter. The perceivable objects were then put in places where the robot could see them while the others were hidden. Object detection was tuned so that all perceivable objects were actually detected and correctly classified.

Table 2 shows the obtained results for three different values of P_{perc}: 0.3, 0.5, and 0.7. We remark that the monitor is able to detect most of the failure situ-

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Table 2
Results of performing crisp *SKEMon* to monitor the two actions:`pick-up` and `enter`. The rows show the ground truth (Success or Failure) while the columns show the result returned by *SKEMon* (Success, Failure, or Unknown).

		$P_{perc} = 0.3$			$P_{perc} = 0.5$			$P_{perc} = 0.7$		
		S	F	U	S	F	U	S	F	U
Navigation	S	20	0	70	22	0	71	27	0	58
	F	0	412	98	0	418	89	0	437	78
Manipulation	S	9	0	101	22	0	87	22	0	81
	F	0	75	315	0	128	263	0	163	234

ations (true negatives) for the navigation actions (80, 82, and 85 %). However, for the pick-up actions, smaller percentages are detected (19, 33, and 41%). The high percentages of the navigation scenario are explained by the fact that most concept definitions are highly constrained, and therefore, the perception of objects as counter evidence is more likely which reduces the number of cases where the monitor declares *unknown*. The definitions of concepts in the manipulation domain involve only a small number of constrains over related objects, so the probability of observing counter evidence is lower. Moreover, as the concept definitions of bowl and glass are the same, all the situations where the expected outcome is picking up a bowl, but the actually picked up object is a glass (and vice versa) are declared as *unknown*.

The results show also that the number of correctly detected successful execution cases is low. Figure 6 shows the rates of true positives (TPR) and true negatives (TNR) for the different types of the expected object. One can observe that all detected success cases for the navigation actions are from runs where the robot successfully moved into either the kitchen or the utility-room. On the other hand, the detected success cases for the manipulation actions are from runs where the expected object to pick up was a cup or a bottle. This is due to the fact that the robot could see objects that were defined to be exclusively related to those types of rooms and containers. For instance, seeing an oven was sufficient to conclude that the current room was a kitchen, while seeing a handle on the picked up container caused that container to be classified as a cup (see the definitions of relations in appendix A). On the other hand, we can observe that the percentage of correctly detected failures is never zero. This brings us to the conclusion that if the SKB contains constraints that uniquely identify classes of objects, crisp *SKEMon* would be able to detect more successful execution cases, and thus a lower number of *unknown* cases.

When the monitoring process takes a credulous approach, all *unknown* results are counted as *success*. Table 3 shows the rates of true positives (TPR) and false positives (FPR) of crisp *SKEMon* using the two approaches: credulous and non-credulous, i.e., treating *unknown* as a separate case. One can note that the credulous approach detects 100% of successful execution cases; but at the same time it reports higher rates of false positives, especially when applied in

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Table 3
The TPR and FPR results, given as percentages, of crisp *SKEMon* for the two scenarios when treating *unknown* as a separate third case, i.e., non-credulous (N-C), and when *unknown* is considered *success*, i.e., credulous approach (C).

		$P_{perc} = 0.3$		$P_{perc} = 0.5$		$P_{perc} = 0.7$	
	Approach	FPR	TPR	FPR	TPR	FPR	TPR
Navigation	N-C	0	22.22	0	23.65	0	31.67
	C	19.21	100	17.55	100	15.14	100
Manipulation	N-C	0	8.18	0	20.18	0	21.35
	C	80.76	100	67.26	100	58.94	100

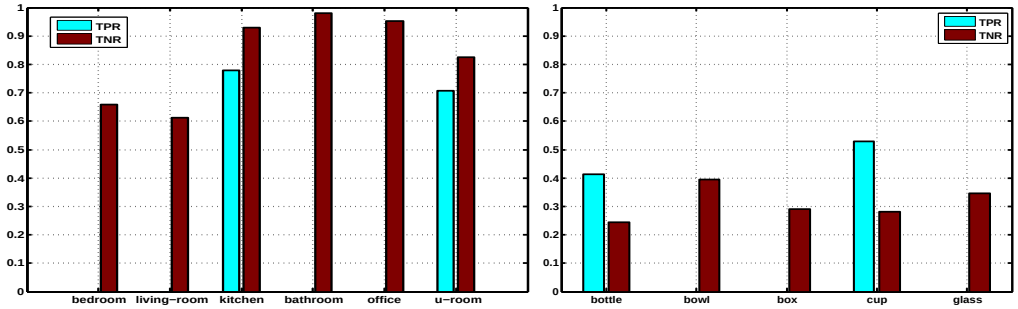

Fig. 6. Rates of true positives (TPR) and true negatives (TNR) achieved by crisp *SKEMon* for different types of rooms (left) and containers (right).

the manipulation domain. It should also be noted that all the results of crisp *SKEMon*, both credulous and non-credulous, indicate a good performance (even for low values of P_{perc}). The reason is that when the FPR vs TPR points are plotted in the ROC space, they are all above the line of no-discrimination (i.e., the line representing the function $f(x) = x$). This result is due to the fact that crisp *SKEMon*, with a non-credulous approach, achieves 100% specificity (which is equivalent to 0% of false positives) and a sensitivity (equivalent to TPR) that is greater than zero. On the other hand, using a credulous approach gives 100% of true positives and a FPR that is less than TPR. Therefore, one can conclude that crisp *SKEMon* is good at detecting when the execution of actions fail, provided that the defined concepts be sufficiently constrained, but is less good in detecting correct execution.

5.2.5 Probabilistic SKEMon

We used the same experimental set-up to evaluate the performance of the probabilistic *SKEMon*. However, this time, we considered action models for *SKEMon* with two possible outcomes each with a given probability of occurrence. For the pick-up action, the first outcome expressed the possibility of picking up the desired object while the second outcome expressed the possibility of picking up another object that was near the desired one. For the navigation scenario, we considered the action `(move loc1 loc2)` to move the robot from its initial room `loc1` to room `loc2`. The first outcome of the action

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Table 4
Results of performance of probabilistic *SKEMon* in monitoring the execution of navigation and manipulation actions. The rows show which outcome of the action actually occurred while columns show the result returned by the *SKEMon* process.

		$P_{perc} = 0.1$		$P_{perc} = 0.3$		$P_{perc} = 0.5$		$P_{perc} = 0.7$	
		O1	O2	O1	O2	O1	O2	O1	O2
Navigation	O1	2337	372	2434	302	2402	302	2446	283
	O2	194	2497	166	2498	144	2552	137	2534
Manipulation	O1	991	832	1100	774	1245	652	1348	554
	O2	551	1376	477	1399	376	1477	340	1508

reflected the situation where the robot would remain unintentionally in `loc1`, while the second outcome expressed the case where the robot would effectively end up in room `loc2`. *SKEMon* was also provided with a sensing model specifying the probabilities for not observing or misclassifying an object.

To simulate the unreliable effects of executing an action, we used the prior probability of the two outcomes, as specified in the action models, to sample the actual object or location. Then, the sampled object was placed in a location where the arm would pick it up, or the robot was placed at the entrance of the sampled room. The process of determining which objects were perceivable by the robot was done the same way as in the crisp *SKEMon* experiments. The acquired perceptual information was corrupted according to the parameters of the sensing model, leading to misclassification.

Probabilistic *SKEMon* was evaluated using four values of P_{perc}: 0.1, 0.3, 0.5, and 0.7. For each value of P_{perc}, three prior probability distributions of the two action outcomes were considered: $(0.8, 0.2)$, $(0.5, 0.5)$, and $(0.2, 0.8)$. We ran experiments where the two objects (involved by the two action outcomes) could be asserted to be of any of the available types. For each combination of types, we repeated the experiment 50 times. This resulted in a total of 3750 runs for the manipulation action and 5400 runs for the navigation action. Table 4 summarizes the results of probabilistic *SKEMon* on both types of actions, i.e., manipulation and navigation. The rows represent the ground truth while the columns show the results of the *SKEMon* process which were computed by selecting the outcome with the higher posterior probability. $O1$, respectively $O2$, refer to outcome 1, respectively outcome 2, of the executed action. The true positive rate (TPR) and the false positive rate (FPR) for probabilistic *SKEMon* were computed by considering outcome 2, i.e., O_2 as the positive case. Table 5 shows TPR vs FPR for both types of actions.

The results indicate good performance as TPR tends to be high and FPR tends to be low. As in the case of crisp *SKEMon*, we notice that the performance gets better when the probability P_{perc} is higher. Similarly, we notice that the performance on the navigation scenario is much better than the one on the manipulation scenario. As explained above, this is due to the number of

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Table 5
The rates of true positives (TPR) and false positives (FPR), given as percentages, of probabilistic *SKEMon* for navigation and manipulation actions.

	$P_{perc} = 0.1$		$P_{perc} = 0.3$		$P_{perc} = 0.5$		$P_{perc} = 0.7$	
	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
Navigation	13.73	92.79	11.03	93.76	11.16	94.65	10.37	94.87
Manipulation	45.63	71.40	41.30	74.57	34.37	79.70	29.12	81.60

constraints and how much of the environment is observable, i.e., P_{perc}. We also ran experiments where the action model used by probabilistic *SKEMon* was corrupted, i.e., the probabilities of the outcomes were not always correct. The obtained results were comparable to those obtained with a reliable action model. These results are reported in the PhD thesis of Bouguerra [35].

6 Discussion and Conclusions

In this paper, we have described a new approach of monitoring the execution of robot plans where semantic knowledge is used to derive and verify implicit expectations of executing plans successfully. We have developed two processes based on this idea. The first process is adequate for domains with deterministic actions and reliable sensing, while the second process was designed to handle domain uncertainty in its different forms, i.e., stochastic action effects, noisy sensing, and incomplete information about world states.

The experimental results show that semantic domain knowledge can effectively help robots achieve good performance of monitoring the execution of their plans. In particular, we have shown that crisp *SKEMon* is very good at detecting execution failures, especially when the knowledge base includes enough counter-evidence constraints. On the down side, crisp *SKEMon* is unable to detect failure situations when the expected object has similar conceptual description as the execution-time object or when the expected object and the execution-time object are both instances of the same class. Although, we did not notice any performance issues regarding the size of the knowledge bases we used in our experiments, we expect that with the use of the state-of-the-art knowledge representation systems, our approach would be able to handle much bigger and more complex knowledge bases. Our claim is supported by the developments of DL-based systems that are shown to scale up well [36,37].

When uncertainty is taken into account, the performance is even better than the one achieved by crisp *SKEMon*. This claim is supported by the TPR and FPR results of monitoring the execution of two different types of actions. The reason for the better performance can be attributed to the fact that expectations are no longer treated in a boolean manner, i.e., satisfied, violated, or unknown. Furthermore, decisions about whether the execution of an action

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

has failed or succeeded are based on how likely each expectation is verified or violated given the acquired perceptual information.

The use of probabilities to model uncertainty in sensing and acting gives us a well founded treatment, but providing the needed probability values might be a daunting task for large domains. In our implementation, we took a Bayesian approach and interpreted probability values as measures of belief. Therefore, the user can model probability state functions using known probability mass functions such as the shifted geometric and Poisson distribution. Using a known probability mass function reduces considerably the amount of information that the user has to provide, since only a few parameters need to be specified. We also simplified the task of providing conditional probabilities for the sensing model by making assumptions that allows us to use well known probability mass functions. The sensing model is relatively complex, and it may be hard to compute for domains with a large number of object types. We have proposed elsewhere a simplified model that assumes no misclassification [6]. Alternatively, one may use approximate inference methods to address the computational complexity of the sensing model: this last point is the subject of our current work. Another point that is worth investigating further is to study how the available semantic knowledge can be used by the perception process in order to focus its attention only on those percepts that are relevant to the task at hand.

While we have evaluated the performance of our approaches in monitoring the execution of simple actions using a simple vision system. We expect that our approaches, especially the probabilistic one, would perform very well in more realistic scenarios when more powerful perception systems are used. This is so, because the probabilistic approach uses a sensing model that is able to reason about noisy sensing where objects can be missed or misclassified when they are detected.

Although, several of the examples and experiments involved the robot's location, the problem addressed by our work should not be confused with self-localization. Our focus is on monitoring the execution of actions by observing their (explicit and implicit) effects: these effects may include the robot's location in the case of navigation actions, but they include other aspects of the world state for other actions, like grasping actions. However, we would like to point out that the output of self-localization can be taken into account when computing the prior probabilities of the outcomes of the action to execute. Therefore, our approach would greatly benefit from using the output of probabilistic localization systems, such as particle-filter ones [38].

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

Acknowledgements

This work was supported by the Swedish KK foundation. We would like to thank the reviewers for their informative and helpful feedback.

References

[1] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[2] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *Int. J. of Robotics Research*, 17(4):315–337, 1998.

[3] M. Beetz. *Plan-Based Control of Robotic Agents*. Number 2554 in Lecture Notes in AI. Springer-Verlag, 2002.

[4] O. Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, 2005.

[5] A. Bouguerra, L. Karlsson, and A. Saffiotti. Semantic knowledge-based execution monitoring for mobile robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 3693–3698, 2007.

[6] A. Bouguerra, L. Karlsson, and A. Saffiotti. Handling uncertainty in semantic-knowledge based execution monitoring. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 437–443, 2007.

[7] S. Lemai and F. Ingrand. Interleaving temporal planning and execution in robotics domains. In *National Conf. on AI*, pages 617–622, 2004.

[8] K. Haigh and M. Veloso. Planning, execution and learning in a robotic agent. In *Int. Conf. on AI Planning Systems*, pages 120–127, 1998.

[9] O. Pettersson, L. Karlsson, and A. Saffiotti. Model-free execution monitoring in behavior-based robotics. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 37(4):890–901, 2007.

[10] G. DeGiacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 453–465, 1998.

[11] M. Fichtner, A. Großmann, and M. Thielscher. Intelligent execution monitoring in dynamic environments. *Fundamenta Informaticae*, 57(2-4):371–392, 2003.

[12] K. BenLamine and F. Kabanza. History checking of temporal fuzzy logic formulas for monitoring behavior-based mobile robots. In *IEEE Int. Conf. Tools with AI*, pages 312–319, 2000.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

[13] F. Ingrand, M. Georgeff, and A. Rao. An architecture for real-time reasoning and system control. *IEEE Expert: Intelligent Systems and Their Applications*, 7(6):34–44, 1992.

[14] R. Firby. Building symbolic primitives with continuous control routines. In *Int. Conf. on AI Planning Systems*, pages 62–69, 1992.

[15] J. Fernández, R. Sanz, and A. Diéguez. Probabilistic models for monitoring and fault diagnosis: Application and evaluation in a mobile robot. *Applied AI*, 18(1):43–67, 2004.

[16] V. Verma, G. Gordon, R. Simmons, and S. Thrun. Particle filters for rover fault diagnosis. *IEEE Robotics & Automation Magazine*, 2004.

[17] G. Fraser, G. Steinbauer, and F. Wotawa. Plan execution in dynamic environments. In *Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, pages 208–217, 2005.

[18] J. Fernández and R. Simmons. Robust execution monitoring for navigation plans. In *IEEE/RSJ Conf. on Intelligent Robots and Systems*, pages 551–557, 1998.

[19] M. Beetz and D. McDermott. Fast probabilistic plan debugging. In *European Conf. on Planning*, pages 77–90, 1997.

[20] C. McCarthy and M. Pollack. Towards focused plan monitoring: A technique and an application to mobile robots. *Autonomous Robots*, 9(1):71–81, 2000.

[21] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, and J. González. Multi-hierarchical semantic maps for mobile robotics. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3492–3497, 2005.

[22] 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 *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1338–1343, 2002.

[23] J. Hois, M. Wünstel, J. Bateman, and T. Röfer. Dialog-based 3D-image recognition using a domain ontology. In *Int. Conf. on Spatial Cognition*, pages 107–126, 2006.

[24] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann. 3D mapping with semantic knowledge. In *RoboCup Int. Symposium*, pages 335–346, 2005.

[25] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[26] R. MacGregor. Retrospective on LOOM. Technical report, Information Sciences Institute, University of Southern California, 1999.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

[27] A. Bouguerra, L. Karlsson, and A. Saffiotti. Active execution monitoring using planning and semantic knowledge. In *ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, 2007.

[28] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 14–15. Prentice Hall, second edition, 2003.

[29] D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *National Conf. on AI*, pages 390–397, 1997.

[30] A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2):481–526, 1995.

[31] A. Ramisa, S. Vasudevan, D. Scaramuzza, R. de Mántaras, and R. Siegwart. A tale of two object recognition methods for mobile robots. In *Int. Conf. on Computer Vision Systems*, pages 353–362, 2008.

[32] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[33] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2149–2154, 2004.

[34] A. Saffiotti and M. Broxvall. PEIS ecologies: Ambient intelligence meets autonomous robotics. In *Int. Conf. on Smart Objects and Ambient Intelligence*, pages 275–280, 2005.

[35] A. Bouguerra. *Robust Execution of Robot Task-Plans: A Knowledge-based Approach*. PhD thesis, Department of Technology, Örebro University, Sweden, 2008.

[36] R. Möller, V. Haarslev, and M. Wessel. On the scalability of description logic instance retrieval. In *Int. Workshop on Description Logics*, 2006.

[37] Y. Guo, A. Qasem, and J. Heflin. Large scale knowledge base systems: An empirical evaluation perspective. In *National Conf. on AI*, 2006.

[38] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2000.

[39] N. Howden, R. Rnnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents – Summary of an Agent Infrastructure. In *Int. Conf. on Aut. Agents*, 2001.

[40] T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

A Appendix: Semantic Knowledge Base

```
;; -------------  Manipulation Domain ---------------
; Relations
(defrelation has-handle :domain cup :range handle)(defrelation has-cap :domain bottle :range cap)
(defrelation has-cover  :domain container :range cover)
;;-------------------------------------------------
; Atomic Concepts
(defconcept handle)(defconcept cover)(defconcept cap)(defconcept container)
;;-------------------------------------------------
; Defined Concepts
(defconcept cup
   :is (and container (exactly 1 has-handle)(exactly 0 has-cover)(exactly 0 has-cap)))
(defconcept glass
   :is (and container (exactly 0 has-handle)(exactly 0 has-cover)(exactly 0 has-cap)))
(defconcept bottle
   :is (and container (exactly 0 has-handle)(exactly 0 has-cover)(exactly 1 has-cap)))
(defconcept box
   :is (and container (exactly 0 has-handle)(exactly 1 has-cover)(exactly 0 has-cap)))
(defconcept bowl :is (and container (exactly 0 has-handle)(exactly 0 has-cover)(exactly 0 has-cap)))
```

```
;; -------------  Navigation Domain ---------------
; Relationships
(defrelation has-oven :domain kitchen :range oven) (defrelation has-bed :domain room :range bed)
(defrelation has-sofa :domain room :range sofa)(defrelation has-table :domain location :range table)
(defrelation has-tv-set :domain room :range tv-set)(defrelation has-fridge :domain room :range fridge)
(defrelation has-pc :domain room :range pc)      (defrelation has-plant :domain location :range plant)
(defrelation has-clothes-dryer :domain room :range clothes-dryer)
(defrelation has-washing-machine :domain utility-room :range washing-machine)
(defrelation has-sink :domain (or kitchen bathroom utility-room) :range sink)
(defrelation has-tub :domain (or bathroom utility-room) :range tub)
(defrelation has-chair :domain location :range chair)
;;-------------------------------------------------
; Atomic Concepts
(defconcept oven)(defconcept bed)(defconcept sofa)(defconcept table)(defconcept washing-machine)
(defconcept sink)(defconcept tub)(defconcept chair)(defconcept fridge)(defconcept tv-set)
(defconcept clothes-dryer)(defconcept plant)(defconcept pc)(defconcept room )(defconcept corridor)
;;-------------------------------------------------
; Defined Concepts
(defset location :is '(corridor room))
(defconcept  bedroom :is
   (and room (at-least 1 has-bed)(at-most 1 has-sofa)(exactly 0 has-sink)(exactly 0 has-oven)
       (exactly 0 has-tub)(exactly 0 has-washing-machine)(exactly 0 has-clothes-dryer)))
(defconcept  living-room :is
   (and room (at-least 1 has-sofa)(exactly 1 has-tv-set)(exactly 0 has-sink)(exactly 0 has-oven)
       (exactly 0 has-tub)(exactly 0 has-washing-machine)(exactly 0 has-clothes-dryer)))
(defconcept kitchen :is
   (and room (at-least 1 has-sink)(exactly 1 has-oven)(at-least 1 has-fridge)(at-least 1 has-table)
       (exactly  0 has-pc)(at-most 1 has-sofa)(exactly  0 has-bed)(exactly 0 has-tub)
       (exactly 0 has-washing-machine )(exactly 0 has-clothes-dryer) ))
(defconcept bathroom :is
   (and room (at-least 1 has-sink)(exactly 1 has-tub)(at-most 2 has-chair)(at-most 1 has-table)
       (exactly 0 has-pc)(exactly  0 has-bed)(exactly 0 has-sofa)(exactly 0 has-fridge)
       (exactly 0 has-oven)(exactly 0 has-washing-machine )))
(defconcept office   :is
   (and room (at-least 1 has-table)(at-least 1 has-chair)(at-least 1 has-pc)(exactly 0 has-bed)
       (at-most 1 has-sofa)(exactly 0 has-fridge)(exactly 0 has-sink)(exactly 0 has-oven)
       (exactly 0 has-tub)(exactly 0 has-washing-machine)(exactly 0 has-clothes-dryer)  ))
(defconcept utility-room :is
   (and room (at-least 1 has-washing-machine)(exactly 1 has-clothes-dryer)(exactly 0 has-oven)
       (exactly 0 has-bed)(exactly 0 has-sofa)(exactly 0 has-pc)(exactly 0 has-fridge)     ))
```


Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.

B Appendix: Parameters Used in Probabilistic *SKEMon*

Parameters of the manipulation domain. The state variables represent the number of handles (S_1), covers (S_2), and caps (S_3). The maximum number of objects that a container can have is always one. Therefore, the values of the state functions $p(s_j|r)$ are either 0 or 1. For the sensing model, the following probability parameters of the multinomial functions were used, such that p_4 is the probability of not seeing the corresponding object:

	p_1	p_4		p_2	p_3	p_4		p_2	p_3	p_4
handle	0.8	0.2	cover	0.6	0.2	0.2	cap	0.2	0.6	0.2

Parameters of the navigation domain. There are thirteen state variables representing the number of objects of furniture items. The domains of such variables were fixed from zero to a certain maximum number as follows:

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}
Object	bed	sofa	sink	oven	table	tv	chair	tub	fridge	plant	pc	c-d	w-m
max #	2	2	2	1	2	1	4	1	1	3	1	1	1

where w-m and c-d stand for washing-machine and clothes-dryer respectively. The values of the state functions were subjective and were set as described in section 4.2 with the exceptions listed in table B.1. For the sensing model, we only considered misclassification of few classes of objects as shown in table B.1. Objects of the other classes were either seen (with probability 0.8) or missed (with probability 0.2).

Table B.1
Parameters used in the experiments. The left table shows state functions, while the right table shows the probabilities of the multinomial functions, such that p_{14} is the probability of not seeing the corresponding object.

	$i=0$	$i=1$	$i=2$
$P(S_1 = i\|\text{bedroom})$	0.0	0.7	0.3
$P(S_2 = i\|\text{living-room})$	0.0	0.6	0.4
$P(S_3 = i\|\text{kitchen})$	0.0	0.4	0.6
$P(S_5 = i\|\text{kitchen})$	0.0	0.8	0.2

bed	$p_1 = 0.8$; $p_2 = 0.1$; $p_{14} = 0.1$
sofa	$p_1 = 0.2$; $p_2 = 0.7$; $p_{14} = 0.1$
sink	$p_3 = 0.7$; $p_4 = 0.1$; $p_{14} = 0.2$
oven	$p_3 = 0.1$; $p_4 = 0.7$; $p_{14} = 0.2$
table	$p_5 = 0.8$; $p_6 = 0.1$; $p_{14} = 0.1$

Cite as: A. Bouguerra, L. Karlsson and A. Saffiotti. Monitoring the Execution of Robot Plans Using Semantic Knowledge. *Robotics and Autonomous Systems* **56**(11):942–954, 2008.