International Journal of Knowledge and Systems Science
Volume 7 • Issue 1 • January-March 2016

Robot Task Planning in Deterministic and Probabilistic Conditions Using Semantic Knowledge Base

Ahmed Abdulhadi Al-Moadhen, Cardiff University, Cardiff, UK
Michael Packianather, School of Engineering, Cardiff University, Cardiff, UK
Rossitza Setchi, School of Engineering, Cardiff University, Cardiff, UK
Renxi Qiu, Department of Computer Science and Technology, University of Bedfordshire, Bedfordshire, UK

ABSTRACT

A new method is proposed to increase the reliability of generating symbolic plans by extending the Semantic-Knowledge Based (SKB) plan generation to take into account the amount of information and uncertainty related to existing objects, their types and properties, as well as their relationships with each other. This approach constructs plans by depending on probabilistic values which are derived from learning statistical relational models such as Markov Logic Networks (MLN). An MLN module is established for probabilistic learning and inference together with semantic information to provide a basis for plausible learning and reasoning services in support of robot task-planning. The MLN module is constructed by using an algorithm to transform the knowledge stored in SKB to types, predicates and formulas which represent the main building block for this module. Following this, the semantic domain knowledge is used to derive implicit expectations of world states and the effects of the action which is nominated for insertion into the task plan. The expectations are matched with MLN output.

KEYWORDS

Deterministic Planning, Markov Logic Network, Probabilistic Planning, Semantic Knowledge, Task-Planning

1. INTRODUCTION

Plan generation by an autonomous robot planner acting in human environments is a complex task as it involves dealing with uncertainty and non-deterministic situations. Indeed, in such situations, a variable number of objects may be relevant to its tasks and these objects may be related in various ways. Uncertainty is a feature related to robots acting in real environments, and may occasionally lead to failure in robot operation. To deal with unexpected planning contingencies, robotic task planning employs probabilistic inference procedures, based on reasoning techniques, to ensure that the generation of its plans does not mean that said plans deviate from their intended course of action (Bouguerra, Karlsson & Saffiotti, 2007). Most approaches to plan generation focus on deterministic information regarding the robot environment, such as the exact objects and their properties, and explicit actions' details (pre-conditions and post-conditions). For instance, the explicit effect of a robot moving to a kitchen is that the robot's new position is the kitchen. In contrast, the implicit effect is that the robot should observe that there is a fridge and an oven.

DOI: 10.4018/IJKSS.2016010104

In a real world environment this is not always realistic, as planning with uncertainty is a complex process. As such, more advanced forms of probabilistic reasoning engaging with semantic knowledge should be proposed to derive probabilistic implicit information which is related to the existence of objects, predicting their types, properties and the relationships among said objects. For example, it is highly likely that a robot moving into a living room would see a TV-set and a sofa, although the probability of the robot encountering a bed and sink is significantly lower.

If the robot is entering a kitchen instead, there is a higher probability that it will see a fridge and a sink. These probabilities are details that would add more complexity to the task planner, if the robot itself is left to process them. In light of this, it is important to build a separate unit based on semantic knowledge and the action descriptions used by the planner. Indeed, the planner has the ability to learn from a knowledge base and then infer probable information about the robot environment by depending on robot observations to support its task planner.

In this paper, the robot planning system infrastructure is based on a semantic knowledge base and represented by Description Logic (DL) (Baader, McGuinness, Nardi & Patel-Schneider, 2010). This type of system has the ability to infer the types of things and the automatic classification of things based on their classes and properties, following which it can develop a general algorithm for this system based on DL. With this, the robot can derive new information from its existing knowledge. Pure description logics inference is completely deterministic, so it is often desirable to represent uncertain information in a way that can be more useful to the robot planner.

A second process has been developed that takes into its consideration probabilistic uncertainty in the state of the world, i.e. uncertainty about the type of objects and places in the robot environment and the relations between them. This process depends on Markov Logic Networks which allow for probabilistic inferences that combine the expressiveness of first-order logics with the representation of uncertainty (Richardson & Domingos, 2006). The influence of such relational structures, involving variable sets of objects on the facts relevant to the robot's tasks, must clearly be omitted from consideration for a model that involves a fixed set of propositions. As such, a unification stage between the principles of first-order logic is crucial, as this makes objects and relations the main building blocks of the representation, and probabilistic graphical models, which enable reasoning in the face of uncertainty, to create a single representation formalism. This approach makes it possible, by combining the respective semantics, to obtain a language that has a level of expressiveness that is adequate for robot communication. Indeed, this is essential if said robots are to be capable of processing uncertain situations when they arise in real-world applications.

This article builds upon and enhances previous work on handling uncertainty using a semantic knowledge base (Al-Moadhen, Packianather, Setchi & Qiu, 2014). In addition, this article reports a new extensive experimental evaluation of two plan generation approaches: a deterministic process and a probabilistic process. These experiments show that the use of semantic knowledge contributes to more intelligent plan generation because the task planner can deal with deterministic and probabilistic situations that might face the robot. They show that the deterministic version mainly operates by finding opposite evidence, while the probabilistic version is better able to take into account true evidence.

The main contributions of this paper are as follows: (i) its semantic knowledge base is used as the main source to construct MLN template; (ii) it uses MLN in cases where there is uncertainty about world states (about objects' existence and their types and relationships); (iii) it devises an algorithm whose role is to create MLN from a semantic knowledge base; (iv) an algorithm, which takes into consideration the semantic of action parameters, is used to check the insertion of action into the plan (in case of deterministic plan generation); (v) in case of a probabilistic version, the plan generation process uses the MLN answers to generate the plan. The MLN template will be learned by using training data from SKB and appropriate learning methods. This learned MLN model is then used to

answer queries generated from the knowledge base and predict the existence and types of objects or places, as well as their interrelationships.

The remainder of this paper is organised as follows. Section 2 describes related works in the field of planning with a focus on deterministic and probabilistic approaches. Section 3 describes planning under deterministic conditions. Section 4 explores planning under uncertainty and introduces the proposed approach to address this issue. Section 5 presents the experimental validation. Finally, Section 6 provides conclusions and suggestions for future work.

2. RELATED WORK

This section reviews research that has addressed plan generation for mobile robot tasks from one point to another. Previous approaches have focused on planning under deterministic world states. In general, predefined world models are used to describe the states of the environment after actions have been correctly executed. Planning under deterministic conditions suffers from a fewer number of issues compared to planning under probabilistic conditions. The planning domains can be generated by integrating semantic action models with a common-sense knowledge base (Al-Moadhen, Qiu, Packianather, Ji & Setchi, 2013).These domains are dependent on fixed information from a knowledge base and input into a robot planner to generate a suitable plan for given tasks.

Semantic knowledge can be successfully used to support robot task planning. Research by (Galindo, Fernández-Madrigal, González & Saffiotti 2008) defined a specific type of semantic map, which integrates hierarchical spatial information and semantic knowledge. This approach is dependent on asserted information and does not take into consideration the uncertainty of robot operation.

Further work by (Eich, Dabrowska & Kirchner 2010) bridged the gap between the semantic and spatial representation of environment representation. Focusing on the semantic side, (Cipriano Galindo et al. 2008) and (Galindo et al. 2005) describe how semantic maps are used for high level planning and spatial reasoning. In their work, the bridging between the spatial domain and the semantic domain is called S-Box (spatial box) and T-Box (taxonomy box). The semantic interpretation of physical objects is achieved by means of optical marker identification, although it is not based directly on spatial interpretation of point cloud data.

Another approach (Bouguerra & Karlsson, 2004) proposed hierarchical task planning that handles uncertainty in both the state of the world and the effect of actions. It introduced mechanisms to handle situations whereby there was incomplete and uncertain information regarding the state of the environment. Indeed, this was done by using belief states to represent incomplete information about the state of the world, while actions were allowed to have more than one outcome. The problem with this approach (using a conditional planner) is its link to conditional branching, which is dependent upon exogenous events, namely, the player interaction.

Partially observable Markov decision processes (POMDPs) (Papadimitriou & Tsitsiklis, 1987) provided a principled general framework for robot motion planning in uncertain and dynamic environments. Furthermore, POMDPs had been used in a study by (Ong & Hsu, 2010) for motion planning under uncertainty. Semantic information can be used as a tool to improve task planning in complex scenarios where other planners may easily find themselves in intractable situations (Galindo, Fernández-madrigal & Saffiotti, 2007). The approach involved constructing a "semantic" plan composed of categories of objects, places, etc. that solved a "generalised" version of the requested task. That plan was then used to discard irrelevant information regarding the definitive planning carried out on the symbolic instances of those elements (that correspond to physical elements of the world within which the robot can operate).

Further work in this area (Galindo, Fernández-Madrigal & González, 2004) included developing a formalism of the environment's symbolic model to solve the issues of processing large amounts of information in planning and being efficient in human–machine communication in a natural form through a human-inspired mechanism that structures knowledge in multiple hierarchies. Planning

with a hierarchical model may be efficient even in cases where the lack of hierarchical information would make it intractable. However, this method was dependent on deterministic information.

In order to deal with uncertainties, probabilistic methods can be used to link the system variables to each other and construct a network among these variables. Markov Logic Networks (Richardson & Domingos, 2006) and Bayesian Logic Networks (Jain, Waldherr & Beetz, 2009) can be utilised to process probabilistic information to support many applications such as task planning.

3. SEMANTIC KNOWLEDGE BASE (SKB) IN PLANNING SYSTEM

Semantic knowledge refers to knowledge about objects, their classes and properties, as well as the relationships between said objects. The representation that refers to this structure is called ontology.

Figure 1 shows the ontology of the robot environment. For instance, a living room is a concept (class) and refers to a room, which has at least a TV-set and a sofa. The entities TV-set and sofa are themselves defined as pieces of furniture.

As is evident from previous research (Al-Moadhen et al., 2013), the semantic knowledge base (SKB) can be used to support robot task planning to generate semantic plans for given tasks. Semantic knowledge is used in the process of generating symbolic plans, i.e. symbols are used to refer to physical objects which will be manipulated by actions. Semantic knowledge can be used by mobile robots to build their maps (Galindo et al., 2008; Tenorth, Kunze, Jain & Beetz, 2010), to analyse their scenes (Hois, Wünstel, Bateman & Röfer, 2007) and to help them communicate with humans (Theobalt et al., 2002).

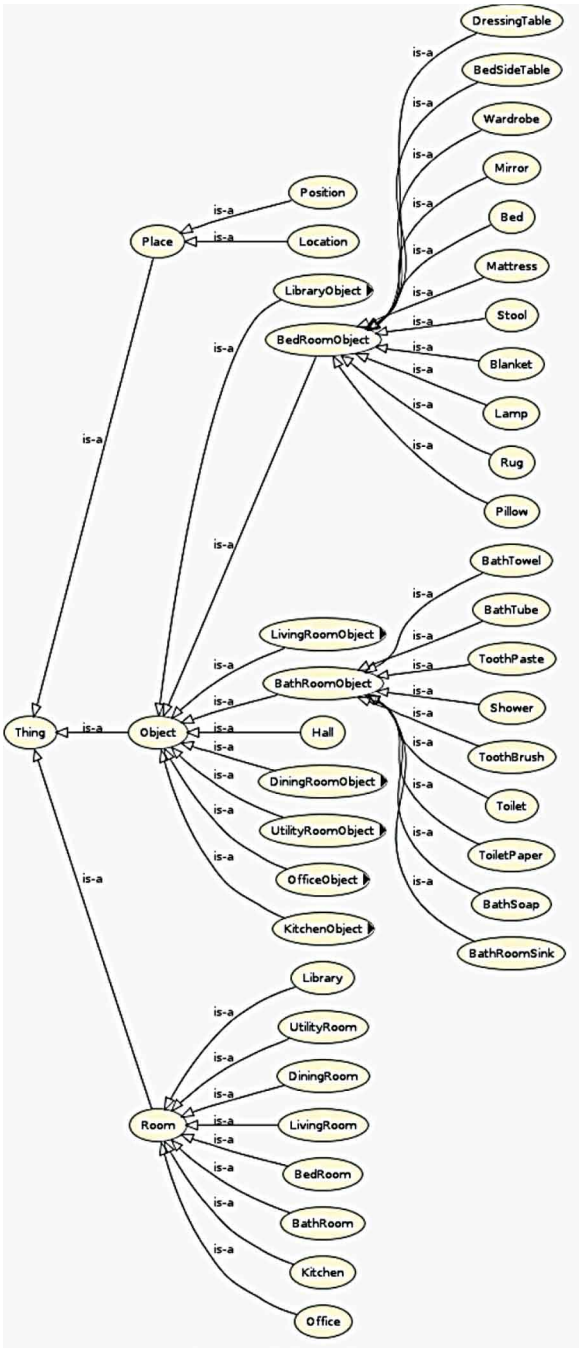
In this paper, semantic knowledge is used during the process of symbolic plans generation, i.e. a sequence of actions that use symbols to refer to physical objects in the robot environment, with these objects related to actions' roles. An example of such an action is (grasp m) where m is a symbol that refers to a milk-box. The robot can use its knowledge to express the milk-box as a container which has to be in a kitchen to know its location. This knowledge base uses the Description Logic (DL) (Baader et al., 2010) to represent and manage the semantic domain knowledge, since it provides a good compromise between representation power and reasoning tractability. An important characteristic of DL is its reasoning capabilities, specifically the inference of implicit knowledge from the explicitly represented knowledge. The knowledge is stored as a Web Ontology Language (OWL) file which is represented as ontology ("OWL Web Ontology Language Reference," 2004).

The different modules involved in generating symbolic plans using semantic knowledge are shown in Figure 2.

The plan generation module constructs plans for a given robot task and also keeps track of the current robot world states and the status of the planner in order to generate queries of the knowledge base in case no plan can be created. It also monitors the current estimated state (by depending on the observations) including, for example, the current expected location of the robot. It is in charge of integrating SKB with Semantic Action Models (SAMs) to produce a planning domain file which is fed into the planner during the planning process (translating the semantic style of SKB and SAMs into planning domain definition language (pddl) format which is suitable input for most of the planner). The Observation/Evidence unit contains observations regarding the current world states stemming from the perception module. These provide the planning system with necessary information about the robot environment. The observation information is expressed in a symbolic form that describes objects and their observable properties (e.g., colour, shape, marks, and relative position).

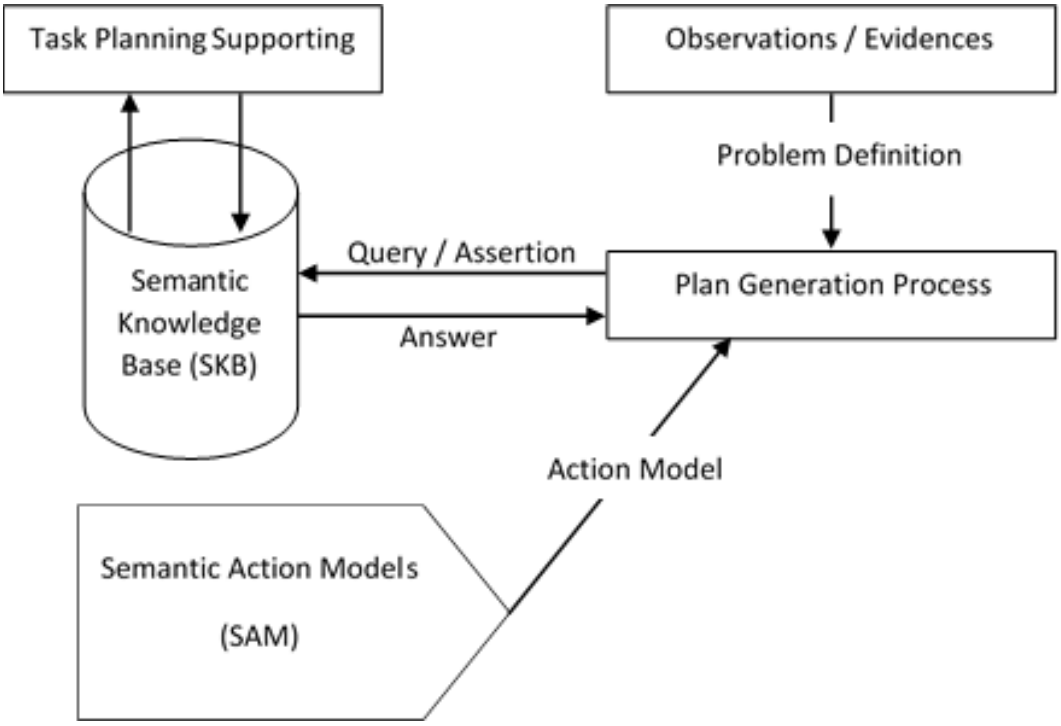
The SAMs module provides the plan generation module with the model of the action which represents the robot capabilities. This model specifies the preconditions that should hold in the world states for the action to be applicable. The model also specifies the explicit effects of the action that should result when the process is executed successfully. The explicit effects are divided into negative and positive effects. Negative effects express the states that should be removed from the world states, while positive effects encode states that should be added to world states when the action is executed

Figure 1. Robot world ontology



successfully. The function of the semantic knowledge base (SKB) is to store general and assertion domain knowledge. It also deals with queries which are generated from plan generation modules and answers these queries according to its contents. Finally, the planning support module itself is in charge of providing SKB with missing information to recover the planning system from situations where no

Figure 2. Different modules involved in planning generation



suitable plans can be created for a given task. It contains methods and procedures for learning and reasoning to infer new information from the current SKB and given observations.

3.1. Robot Environment Ontology

A large amount of knowledge regarding objects and their properties in a robot environment can assist the robot to learn about the structure of its environment. This knowledge can be represented as an ontology that contains classes of all the relevant types of objects and their properties and relations.

Description Logic (DL) is used as a formalism to represent the robot's knowledge of its world. The Web Ontology Language (OWL) ("OWL Web Ontology Language Reference," 2004) has a knowledge representation format suitable for describing a robot's knowledge of its environment. It is used for storing Description Logic formulas in an XML-based file format. There are two types of knowledge in Description Logic. The first is called terminological knowledge, the so-called TBOX, while the second one is known as assertion knowledge, the ABOX. The TBOX contains definitions of concepts, for example the concepts Room, Action or Object. These concepts are arranged in a hierarchical fashion, which is classified using subclass definitions that describe, for instance, a table as a subclass of furniture. The ABOX contains individuals which are instantiated from TBOX concepts, for example, table1 is an instantiation of the concept table. Individuals' properties in the knowledge base can be described by roles which reflect the relations between two individuals, and can also be used in class definitions to constrain the extent of a class to individuals having certain properties e.g. the concept GrasspingABottle can be described as a subclass of GraspSomething with the restriction that the objectActedOn has to be some instance of Bottle.

The following examples (which are based on Figure 1) describe the definition of atomic classes and defined classes. The roles are used to restrict the extension of certain classes.

Table 1. Results of a deterministic plan generation of the actions pick-up and move

		Th = 3 objects			Th = 5 objects			Th = 7 objects		
		M	**U**	**A**	**M**	**U**	**A**	**M**	**U**	**A**
Navigation	**M**	3	0	10	4	0	10	5	0	8
	U	0	55	12	0	56	10	0	60	7
Manipulation	**M**	1	0	8	2	0	7	2	0	6
	U	0	6	25	0	10	21	0	13	19

Atomic classes: Room, Object.
Defined classes: Kitchen isA Room and contains KitchenObject
KitchenObject isA Object and isLocatedAt Kitchen

The arrangement of the ontology levels, consisting of many classes, their hierarchy and properties, has been adopted from the OpenCyc ontology (Lenat, 1995) and KNOWROB (Mori Tenorth & Beetz, 2009) and this adaptation describes the way in which things can be explained.

Figure 1 shows the hierarchical levels of the robot environment ontology. The most important branches are the Rooms, containing descriptions of room types, and Objects, containing descriptions of object types. Most objects in the robot's environment, together with pieces of furniture or body parts, are subsumed under the Object class. Other notable branches include the Place class, which describes locations and positions in the robot map.

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

```
SELECT ?k1 ?f1
WHERE        {?k1 rdf:type Kitchen
       ?f1 rdf:type Object
          ?k1 contains ?f1}
```

asserts that k1 is an instance of kitchen and f1 as an instance of object. As a result of the property "contains" between these two instances, f1 is classified as one of the kitchen objects. The instance k1 is also classified automatically as an instance of the class room. Classification is performed based on the definitions of concepts and relations to create a domain-specific hierarchy. The hierarchy 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 according to what has been asserted.

3.2. Semantic Plan Generation Process

A process for semantic knowledge based plan generation is outlined in algorithm 1. The process typically checks if a planning time object fits the description of an expected object. For instance, if the action to be inserted into the plan is grasp b1 to grasp object b1 (the expected object), then the object that will actually be manipulated by the action grasp is the planning time object; indeed, this needs to be checked to verify if it matches the description of b1. The appending of the navigation action (move r1 r2) implies that the planning time object is the room where the robot should end up, which needs to be checked against the description of r2 (the expected object).

The process begins by establishing the type of the expected object (obj), which comes from the semantic action model; (obj) is derived from the positive effects of the selected action. Following this, the process continues by querying the SKB about the assured classes of the expected object (obj) (step 1). Only the most specific classes are considered, since the semantic knowledge base can deduce

Algorithm 1. The pseudo code of the semantic knowledge-based plan generation process

```
PlanGen(obj)
1. classes  <== SKB[type(obj)]
2. temp-obj  <==Evidence[obj]
3. α <== Evidence [properties-and-relations(temp-obj)]
4. create- instance-of (temp-obj, α)
5. if  ∀ c ∈ classes: is-instance-of(temp-obj, c) then success
6. else if ∃ c ∈ classes: is-not-instance-of(temp, c) then failure
7. else ambiguous outcome
End
```

that an instance of a specific class is also an instance of all the more general classes. For example, if r2 is assured to be a room and a kitchen, then only the kitchen class is considered.

In step 2, the observation/evidence unit is asked to return the planning-time object which is given a temporary name by this process. The evidence unit is then queried in terms of the related properties and relations according to the other observed objects of the planning-time object (step 3).

In step 4, the observed information relating to the planning-time object is used to construct a temporary instance in SKB. For example, if the observation database has the element of room type which contains chair ch1 and bed b1, then the planning process verifies those facts in the SKB by issuing the following SPARQL command:

```
SELECT ?temp ?ch1 ?b1
WHERE {?temp rdf:type Room
       ?ch1 rdf:type Chair
       ?b1 rdf:type Bed
        ?temp contains ?ch1
        ?temp contains ?b1
      }
ASK  { ? temp rdfs:subClassOf  LivingRoom }
ASK  NOT EXISTS { ? temp rdfs:subClassOf  LivingRoom }
```

3.3. Deterministic Planning System Process

The planning method described in (Al-Moadhen et al. 2013) deals with deterministic information stored in a semantic knowledge base. It is used as static information by the planner to generate its plans. The next section extends this planning approach by supporting the planning system with probabilistic capabilities.

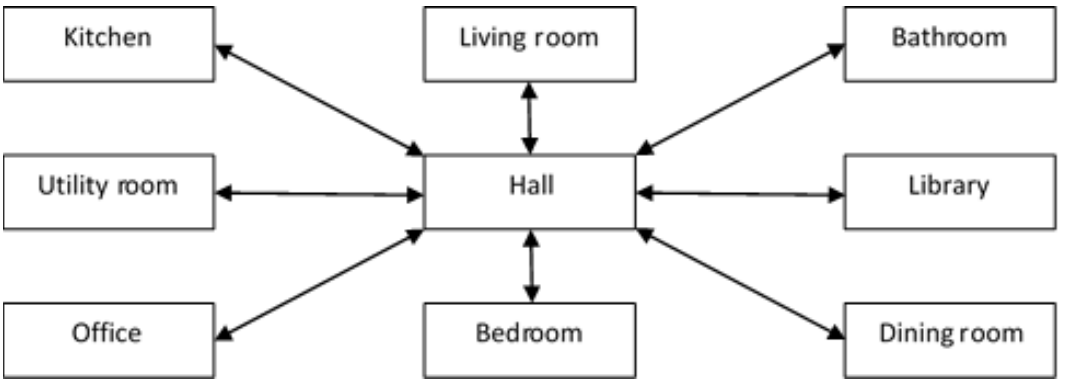
This semantic planning process is based on running a given planner, such as, for instance, Metric-FF (Hoffmann & Nebel, 2001), using the information stored in the semantic knowledge base to improve planning efficiency in relational models that contain a high number of objects and/or relations.

The plan generation module receives as input a specification of the goal to achieve, specified using symbols of the spatial information and objects in the robot environment.

The planning process consists of the following steps:

1. A task is requested for planning, e.g. “bring me milkbox”. This task is specified as a goal to achieve (a world state that has to be reached) using only symbols, i.e. milkbox-1.
2. All the symbols in the goal are translated through the semantic definition, thus the task “bring me milkbox” is translated into “bring MILKBOX”.
3. For the “bring me milkbox” goal, considering the semantic network depicted in Figure 3 and assuming that the robot is in the livingroom, the constructed plan would be: “move from LIVINGROOM to HALL”, move from HALL to KITCHEN”, “Open FRIDGE” and “Pickup MILKBOX”. If a plan exists then the task is solved, otherwise the task must be solved using the nondeterministic task planning approach. This approach will be explained in the next section.

Figure 3. Example of the lowest level of a semantic hierarchy for a house scenario. Arrows indicate "isConnectedTo" relationships house rooms.



4. All the symbols in the robot ontology that do not correspond to classes involved in the semantic plan are discarded for the subsequent planning process. In the above example, only distinctive places, namely the kitchen, the livingroom, fridge and milkbox are considered, with the other elements of the domain discarded.

4. PLANNING UNDER UNCERTAINTY

The plan generation process presented in the previous section is limited to deterministic world states. The process also has the limitation of treating the world expectations in a boolean manner, i.e., evaluated as being either true, false, or unknown. These limitations stem mainly from the fact that the process does not represent uncertainty inherent in world states.

This section describes a different plan generation process that is able to reason about uncertainty. Indeed, a model that takes into account quantitative uncertainty in the form of probabilities in states has been developed and the semantic knowledge is used to interpret expectations.

More specifically, world states can encode different possible outcomes, each with a given probability of occurrence, depending on action effects and knowledge base. 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 planning process can compute a probability for whether a certain expectation of the world state is verified. An example would be "the robot is in the bedroom with a probability of 0.90" instead of returning "unknown" as a planning result. Moreover, the fact that a posteriori probability of each possible world state can be estimated enables a more informed decision about how to proceed by considering plan generated successful, failed, or missing information. Indeed, this is superior to the boolean (true or false) approach. The probabilistic planning process works as follows:

- For each possible world state outcome whose related objects and places are involved in the planning process, a set of implicit expectations is determined.
- Those expectations are used to estimate a probability distribution over the actual world state. For instance, the implicit expectation of finding an oven in the kitchen implies that the probability of having a bed is zero, while the probability of having one, two, or more beds in a bedroom is strictly greater than zero. Although reasoning under uncertainty in description logics is an ongoing research activity (Lukasiewicz, 2008), unfortunately there is no available DL system

that supports probabilities. Thus, the probability distributions of the expected state of the world are computed by statistical relational models such as MLN.

Besides uncertainty about the world state, uncertainty in observation 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 observed model permits:

- To state whether or not an object that exists in the real world is seen.
- How a seen object is classified, i.e., the model accounts for misclassification of objects when they are seen. For instance, a mistake may happen when a bed is classified as a sofa.

Semantic knowledge base is proposed to support the robot planning system by handling the issues of uncertainty regarding the existence of objects in a certain place or predicting types of objects or places, as well as the relationships between them. For instance, to insert an action such as (move bedroom1 kitchen1) into the robot plan, it might be necessary to provide the robot planner with descriptions of the next room that the robot should move to. If the task requires the robot to fetch a milk box and there is no assertion about the milk box location, it is necessary for the planner to get support from the probabilistic module to provide it with the most probable location of the milk box. If the probabilistic module answers, with high probability, that the most probable location of the milk box is the kitchen because it is a kitchen object, then the best next location in the move action is the kitchen. Planning involving uncertainty in predicting the types of objects or places is conducted in the space of weights associated with a formula that describes them.

In order to support the task planner efficiently, probabilistic models are needed to represent the probability distributions of uncertain information in the planning domain. There are several statistical relational models represented as extensions of undirected (Richardson & Domingos, 2006) or directed graphical models (Multi-Entity Bayesian Networks (Laskey, 2008) or Bayesian Logic networks (Jain et al., 2009). These models can make use of learning and inference methods which have been developed for their underlying representations. Markov Logic Networks are used in this paper as a statistical relational model to support the task planning process. Figure 4 represents the probabilistic planning system architecture proposed in this research.

The planning process uses the prior probability distribution, over the possible status of the world states, together with the semantic knowledge-based probability estimates and the observed model to compute the posterior probability of the status.

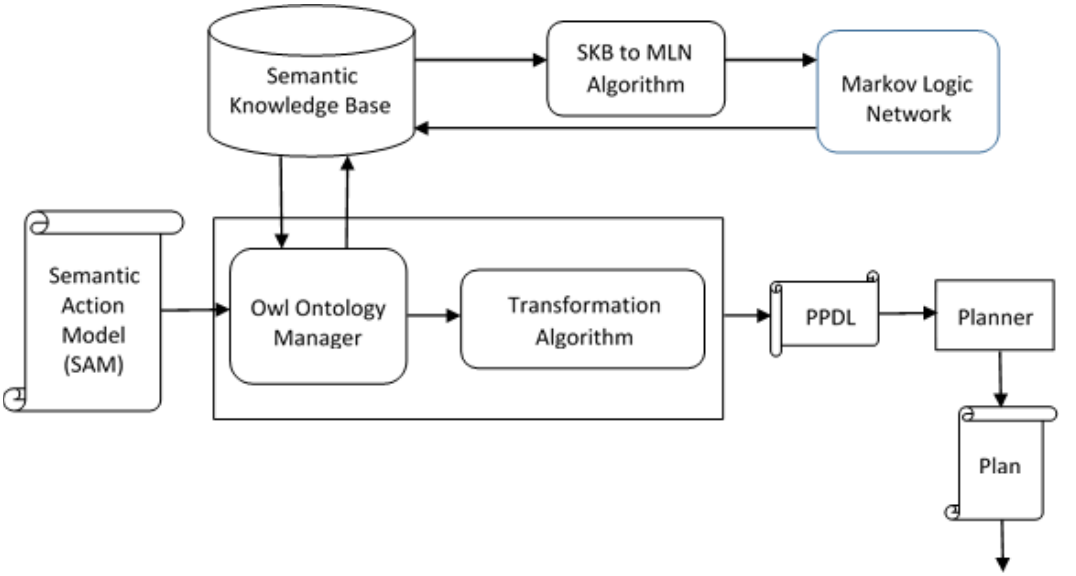
The method brings with it constraints over atomic classes of observable objects. One could easily add constraints over values of other attributes, e.g., colour $\in$ {red, yellow, white}.

Example: Assume that the planning system needs to append the navigation action (move loc1 loc2) whose world state has two possible statuses. The first status, i.e., S =1, is when the robot remains unintentionally in loc1, while the second status, i.e., S = 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 S1 and S2 respectively state whether there are actual beds and sinks that exist in one of loc1 or loc2. O1 and O2 respectively state the observed beds and sinks in the current location.

4.1. Markov Logic Networks (MLNs)

The formal definition of MLN L is given by a set of pairs <Fi, wi>, where Fi is a formula in first-order logic and wi is a real-valued weight. For each finite domain of constants D, an MLN L defines a ground Markov network $M_{L,D}$ = <X,G> as follows (see (Jain 2011) for more details):

Figure 4. Probabilistic planning system



- X is a set of boolean variables. For each possible grounding of each predicate appearing in L, add a boolean variable (ground atom) to X.

  *G is a set of weighted ground formulas, i.e. a set of pairs <Fj', wj'>, where Fj' is a ground formula and wj'is a real-valued weight.

The ground Markov network $M_{L,D}$ specifies a probability distribution over the set of possible worlds X as follows:

$$P(X = x) = \frac{1}{Z}\exp\left(\sum_{i=1}^{|L|} w_i n_i(x)\right) = \frac{1}{Z}\exp\left(\sum_{j=1}^{|G|} w'_j f'_j(x)\right) \quad (1)$$

$$Z = \sum_{x' \in X} \exp\left(\sum_i w_i n_i(x')\right) = \sum_{x' \in X} \exp\left(\sum_j w'_j f'_j(x)\right) \quad (2)$$

4.2. Learning MLN

The learning of a statistical relational model involves the construction of a model from observed training data. The structure of the model can either be known a priori, leaving only the parameters to be determined, as shown in (Kok & Domingos, 2005), or can be part of the learning problem. One consequently differentiates parameter learning from the harder problem of structure learning. The first approach towards learning the structure of MLNs was presented by (Kok & Domingos 2005), while structure learning is clearly important if Artificial Intelligence (AI) systems are to build up probabilistic models with as little human assistance as possible. As such, parameter learning is the most important aspect for knowledge engineers who typically qualitatively assess the properties of

a distribution and indicate the dependencies between the variables but cannot quantitatively define the degree to which these variables depend on each other.

In a Markov logic network, the goal of parameter learning is to set the weights of the model's formulas such that they reflect observations that have been made about the particular part of the world with which the model is concerned. The observations that were made are representative of the particular aspects of the world that are to be captured by the model, such that they allow the model to extract precisely the general principles. The observations used for learning can be stored in a training database that uses the same language as the model.

Since MLNs use logical predicates, the database should thus contain the truth values of a number of ground atoms. The entities appearing in the training database implicitly define a set X of ground atoms. Any ground atoms in X whose truth values are not given in the training database are assumed to be false (closed world assumption). Under this assumption, the training database thus specifies a full assignment X = x.

Maximum Pseudo-Likelihood Learning: This method can be addressed by maximising Pseudo-Likelihood.

$$P\left(y|x\right)=\frac{1}{Z_x}\exp(\sum_{i\in F_Y}w_i n_i\left(x,y\right)) \tag{3}$$

where X_k is a ground atom, x_k is $X_{k's}$ truth value in x, and $MB_x(X_k)$ is the assignment of $X_{k's}$ Markov blanket in x (Richardson & Domingos, 2006). The pseudo-likelihood approximates $P(X = x)$ by making strong independence assumptions, thus avoiding an expensive inference process.

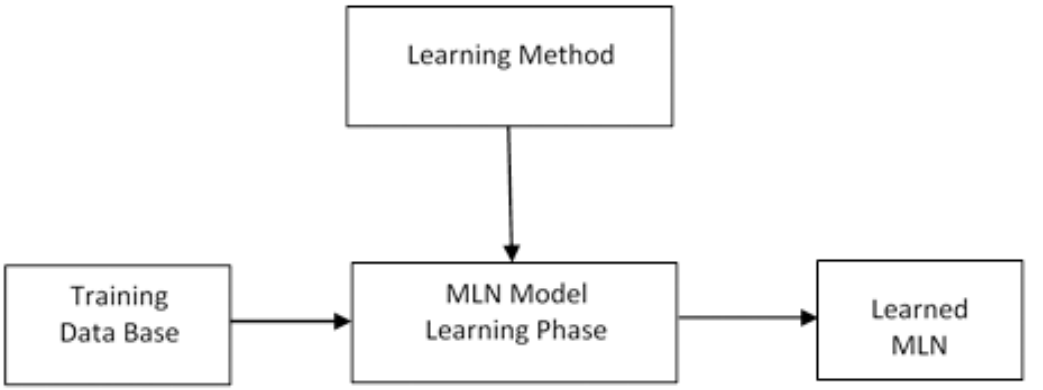
Discriminative Learning: The most attractive feature of undirected models, and therefore MLNs, is that they can also be trained discriminatively, i.e. they can be trained to represent not a full-joint distribution but a conditional distribution. Assuming that there is a strict separation between observable and unobservable variables in the application at hand (e.g. a classification task, where only the classes are always unknown but everything else is given), discriminative training can yield models with superior performance. Methods for the discriminative training of MLNs were introduced by (Singla & Domingos 2005). Figure 5 explains the components of the MLN learning phase.

In many applications, *a priori* which predicates will be evidence and which ones will be queried are known, and the goal is to correctly predict the latter given the former. If the ground atoms in the domain are partitioned into a set of evidence atoms X and a set of query atoms Y, the *conditional likelihood* of Y given X is:

$$=\frac{1}{Z_x}\exp\left(\sum_{j\in G_Y}w_i g_i\left(x,y\right)\right) \tag{4}$$

$$\frac{\partial}{\partial w_i}\log P_w\left(y|x\right)=n_i\left(x,y\right)-\sum_{y'}P_w(y'|x)n_i\left(x,y'\right)=n_i\left(x,y\right)-E_w\left[\boldsymbol{n}_i\left(x,y\right)\right]$$

Figure 5. MLN learning phase



where F_Y is the set of all MLN clauses with at least one grounding involving a query atom, $n_i(x, y)$ is the number of true groundings of the i_{th} clause involving query atoms, G_Y is the set of ground clauses in $M_{L,C}$ involving query atoms, and $g_j(x, y) = 1$ if the j_{th} ground clause is true in the data and 0 otherwise. When some variables are hidden (i.e., neither query nor evidence) the conditional likelihood should be computed by summing them out, although for simplicity's sake, all non-evidence variables are treated as query variables. The gradient of the conditional log-likelihood (CLL) is:

$$FPR = \frac{\text{The number of negative instances that are erroneously classified as positive}}{\text{The total number of actual negative instances}} = \frac{\text{FP}}{\text{N}} \tag{5}$$

4.3. MLN as Inference Engine

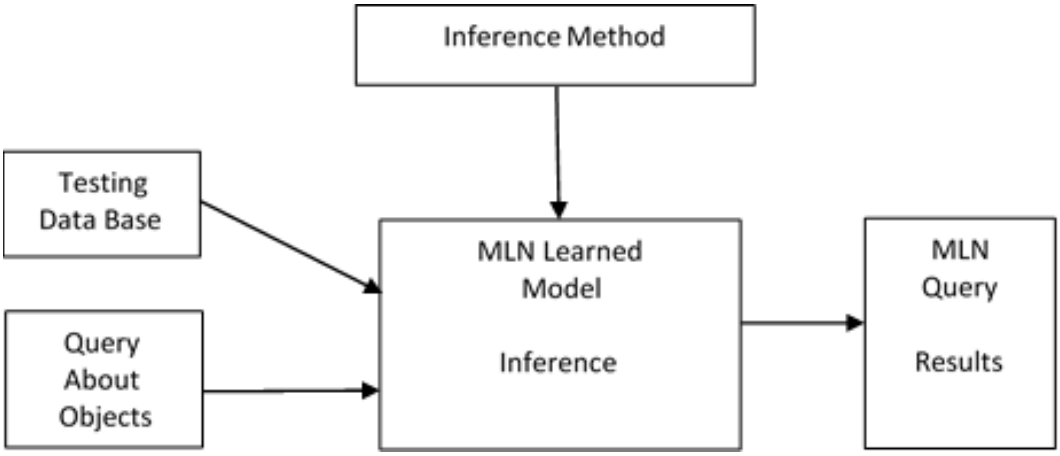
The probabilistic semantics of Markov logic networks are defined via ground Markov random fields, so inference can essentially be handled by applying standard inference methods for Markov networks.

Markov chain Monte Carlo (MCMC) (Koller & Friedman, 2009) is essentially an approximate, sampling-based method where the individual samples are not drawn independently but are taken from a Markov chain, i.e. a model describing sequences of states. Figure 6 explains the components of the MLN query phase.

Formally, a Markov chain over a state space X is given by an initial state distribution π_0: $X \in [0, 1]$ and a probabilistic transition model that makes the first-order Markov assumption: For any given state $x \in X$, the probability distribution over successor states x_0 is given by $T(x \rightarrow x') := P(x' \mid x)$.

MC-SAT applies slice sampling to Markov logic, using SampleSAT to sample a new state given the auxiliary variables. In the Markov network obtained by applying an MLN to a set of constants, each ground clause c_k corresponds to the potential function $\varphi_k(x) = \exp(w_k f_k(x))$, which has value e^{wk} if c_k is satisfied, and 1 otherwise. Please see Poon & Domingos (2006) for more details.

Figure 6. MLN inference (query) phase



4.4. SKB to MLN Algorithm

Algorithm 2 is developed to create MLN from SKB is as follows:

4.5. Examples

1. Suppose that the planner has to insert the movement action (move r2 r1) (where r2 is a livingroom and r1 may be a bedroom or an office), so there are two possible values for r1. Then assume that the r1 value depends on the result of the query that is returned from the learned MLN model. If the result shows that r1 could be a bedroom with probability of 0.2 or an office with probability of 0.8, then the move action should be move r2 r1 with r1 = office.
2. Continuing the example above (move r2 r1), the implicit expectations of being in r2 are determined based on the type of r2. If r2 is asserted to be a bedroom and bedrooms are defined as rooms having a bed (S1) and no sink (S2), then the implicit expectations could be E1 = having bed and E2 = having no sink. The conditional probabilities of the state variables given that the robot is in a bedroom might be determined as follows: having a bed, i.e., S1 in a bedroom can be P(S1 = true|bedroom) = 1, while is the probability of having no sink in a bedroom P(S2 = 0| bedroom) = 1.

5. TESTING SCENARIO

This section will present experiments in simulation to collect large amounts of data for the purpose of statistically evaluating the performance of the proposed framework. Due to a lack of benchmark systems in generating symbolic plans, the evaluation is based 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).

The testing scenario involves performing a navigation and manipulation tasks in a house environment. This house consists of nine rooms, namely kitchen, office, bedroom, bathroom, library, utilityroom, livingroom, hall and diningroom as shown in Figure 7. These rooms have not yet been identified and the robot depends on the MLN module to identify them.

Algorithm 2. Creating MLN from SKB

```
Input: SKB Classes (C), Properties (P), Roles (R)
Output: MLN
Types=null
Domain = null
Predicate = null
Formulas = null
For every c in C
     If c is atomic class
               Type = Type + c
               Domain = Domain + Individuals(c)
For every p in P
     Parameter = (Domain (p), Range (p))
     Predicate = Predicate + p(Parameter)
For every r in R
     Formulas = Formulas + Create formula (r)
MLN = combine (Predicate + Formulas)
Return MLN
```

5.1. Simulation Results

In this section, the Probabilistic Cognition for Technical Systems (ProbCog) ("Intelligent Autonomous Systems - ProbCog Toolbox," 2013) software was used to train the MLN model and to infer new information from it in the experiments. The data of plan generation of manipulation and navigation actions inside a house environment has been collected. A planner is known as Metric –FF (Hoffmann & Nebel, 2001). This planner has become very popular during the past few years. The operation in this paper is not limited to this planner, but any STRIPS-style one can be used. The Metric-FF planner may already have some other techniques for improving its computational efficiency of planning, so the overall improvement of the proposed framework can be high.

5.2. Metrics for Performance Evaluation

The performance evaluation of the proposed methods in this paper is based on the metrics true positive rate (TPR), and false positive rate (FPR). The formulas associated with each metric are as follows:

- $TPR = \dfrac{\text{The number of positive instances that are correctly classified as positive}}{\text{The total number of actual positive instances}} = \dfrac{\text{TP}}{\text{P}}$
- $TPR = \dfrac{\text{The number of positive instances that are correctly classified as positive}}{\text{The total number of actual positive instances}} = \dfrac{\text{TP}}{\text{P}}$

5.3. Manipulation Scenario

In the first scenario, the robot will pick up objects inside the oven and place them on the base of the robot so they can be carried to another location inside the house. There is a general class object to which all of the object's subclasses belong. Some objects have specific properties in terms of restrictions over relations to other atomic objects such as handle. In this environment, there are 18 objects, 4 of which can be picked.

5.4. Navigation Scenario

In this scenario, the robot is acting in a house environment that comprises 9 rooms of different types (kitchen, office, etc.). In each room there are furniture items that are typical for that type of room.

Figure 7. Robot environment

For instance, in a kitchen, there is an oven, sink, etc. In total, there are 18 different types of objects that can exist in a room. The semantic knowledge used in this scenario is more complex than in the previous scenario. The definitions contain more restrictions, and there are more related objects to take into account in order to classify a room. Indeed, there are certain types of objects that, when observed, do not affect the classification process, e.g. lights.

5.5. Threshold for Decision Making

The number of related objects to the room or other objects is specified by the variable (Th). The degree of selecting the desired predicates is dependent on standard deviation (std_{dev}) that can be calculated from the answers derived from the inference engine depicted in Figure 6. There are three types of answer: success when the probability associated with each predicate is over (mean + std_{dev}), failure when the probability associated with each predicate is below (mean- std_{dev}) and unknown when the probability associated with each predicate is between them.

5.6. Deterministic Plan Generation

The performance of the deterministic plan generation process was tested in both scenarios. For the manipulation scenario, each experiment consisted of generating 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:

The planning generation process was called once the planner process reported that it had succeeded in generating (pickup obj) action. The camera on the robot was used to acquire observed information about the picked up object which was then stored in the evidence database.

The "(pickup obj)" experiment was run such that obj was asserted 10 times to be one of each of the four types: cup, bowl, glass water, and milk box, giving a total of 40 runs. In each run, the type of object that was picked up was correctly selected from the 4 available types depending on its descriptions in the evidence database.

Similarly, the navigation scenario consists of the action *(move from to)* to move the robot from hall room to other room identified by the symbol (*to*) and whose type was asserted to be one of the available room types, i.e., bedroom, livingroom, etc. Each room type was considered 10 times, so the total number of runs is 80. For each run, the type of the final location of the robot was selected successfully from the 8 available types. A world state, containing objects that were consistent with the actual location, was then generated using the same threshold from Section 5.5. The objects that could be observed in the robot's observation database were determined according to the threshold parameter. The observable objects were added to its observation database while the others were hidden.

Table 1 shows the earned results for three different values of (Th). The rows of the table represent the matched results between the expected object (obtained from action model in the plan) with the actual (evidence) object (obtained from evidence database). The first row for each scenario represents the positive matching result, i.e., the runs where the expected outcome of the action is the same as the actual outcome. The second row represents negative matching result, i.e., runs where the expected outcome of the action is different from the actual outcome. The columns, on the other hand, represent the results of checking the implicit expectations of the expected object, so the results are either **M**atched, **U**nmatched or **A**mbiguous.

In these results, the deterministic planning is able to show a high percent of true positives (positive instances that correctly classified) and a zero percent of false positives (negative instances that erroneously classified as positive). The true negatives (failure situations) are detected with different percentage between navigation and manipulation scenarios. They are high in navigation (82%, 85%, 90%) and small in manipulation (20%, 32%, 40%) (Table 2).

The difference in failure detection between these scenarios comes from the difference in concept definitions in each scenario. In the case of manipulation, a small number of restrictions are used to define each concept in this scenario, so a large number of conditions will be treated as a negative evidence and ambiguous results will be high. Whereas in case of navigation, the concepts in this scenario are highly constrained, so a few numbers of conditions will be treated as negative evidence and ambiguous results are low.

5.7. Probabilistic Plan Generation

In this section, the same experimental set-up is used to evaluate the performance of the probabilistic semantic knowledge planning system. However, the world states for a system have more than one possible state, each with a given probability of occurrence. For the pick-up action, the first world state expressed the possibility of picking up the desired object while the other state expressed the possibility of picking up another object. For the navigation scenario, the action is considered (move from to) to move the robot from its initial room *from* to room *to*. The first state reflected the situation where the robot would remain unintentionally in *from*, while the other state expressed the case where the robot would effectively end up in room *to*. Probabilistic planning was also provided, with an observed model specifying the probabilities of not observing or misclassifying an object (the predicates that have minus signs before them).

The probability of selecting the desired predicates from the learned MLN is depended on standard deviation (Std_{dev}), which is a statistic used as a measure of the dispersion or variation in a distribution. Std_{dev} is calculated from the weights associated with all the formulas in MLN model. Then, it is used with mean value of the formula weights in the comparison with the value associated with the answer

about queried object in the action related to the current plan. This answer is derived from the MLN model in inference engine phase as depicted in Figure 6. There are three types of answer: *matched* (M) when the probability associated with a predicate is over (mean + std_{dev}), unmatched (U) when the probability associated with a predicate is below (mean- std_{dev}) or ambiguous when the probability associated with a predicate between (mean + std_{dev}) and (mean- std_{dev}). In the case of ambiguous, it needed more evidences (either positive or negative) in order to be (M) or (U). There are 3 cases considered to obtain the value of (Std_{dev}) which are depended on the variable (Th). This variable is represented as the number of related objects in the room or in relation to other objects. The considered cases are when the number of related objects are 3, 5 and 7 objects.

The experiments were run where the two objects (involved in the action outcomes) could be asserted to be of any of the available types. The action model for both scenarios has two possible outcomes with their associated probabilities. These probabilities represent the possible effects of the action on the next generated world state. The probable outcomes of the actions are represented as out_1 and out_2. The out_1 represents the incorrect action effect while out_2 represents the correct action effect. Two probability distributions relate to these outcomes are considered: for the first distribution p(out_1) = 0.3, p(out_2) = 0.7, whereas the second distribution p(out_1) = 0.7, p(out_2) = 0.3. For example, if the probability distributions of action outcome of *(move r1r2)* to move the robot from livingroom (r1) to kitchen (r2) are p(out_1) = 0.3 and p(out_2) = 0.7, so the effect of the action is: the robot at the kitchen. This effect will be compared with the answer returned from the MLN when the planning system asks MLN about the type of r2. If it is matched, then the planning system will continue to generate the other actions to accomplish the task. If it is unmatched, then the other possible outcome will be considered and the same sequence of checking will be started.

For each combination of object or room types and the possible action effects probability distributions the experiment was repeated 10 times, so the total number of runs is 320 for both manipulation and navigation.

Table 3 summarizes the results of probabilistic planning of both types of actions, i.e., manipulation and navigation. The rows represent the selected action outcome with the highest probability in action model, while the columns show the results of the MLN answers which were computed by selecting the answer with the highest weight from MLN. Then the answer values are **M**atched, **U**nmatched or **A**mbiguous.

The true positive rate (TPR) and the false positive rate (FPR) for probabilistic planning were computed by considering outcome 2, i.e., out_2 as the positive case and out_1 as negative case. Table 4 shows TPR vs FPR for both types of action. The results indicate good performance, as TPR tends to be high and FPR tends to be low. As in the case of deterministic planning, it can be noticed that the performance gets better when the probability (Th) is higher. Similarly, one can notice that the performance in the navigation scenario is much better than the one in the manipulation scenario. This is due to the number of constraints and how many related objects in the environment.

By comparing table 2 and table 4, it can be noticed that the deterministic planning system has high (100%) performance in classifying all the correct matching (TPR) between the expected object and the actual (evidence) object as positive. But it has high percent of error when classifying the unmatched cases as positives (FPR), especially in manipulation scenario. This is due to a limited number of restrictions in the definition of objects in that scenario.

On the other hand, the probabilistic planning system has less percent (from deterministic planning) in classifying correct matching as positive (TPR), but it has higher percent (from deterministic planning) in detecting the failure cases (FPR).

Table 3. Results of performance for the probabilistic plan generation of navigation and manipulation actions

		Th = 3 Objects			Th = 5 Objects			Th = 7 Objects		
		U	M	A	U	M	A	U	M	A
Navigation	out_1	120	7	5	128	5	3	132	2	2
	out_2	16	160	12	12	163	9	4	171	9
Manipulation	out_1	100	39	13	112	30	10	124	16	8
	out_2	56	102	10	44	117	7	32	135	5

Table 4. The rates of true positives (TPR) and false positives (FPR), given as percentages, of probabilistic planning system for navigation and manipulation actions

	Th = 3 Objects		Th = 5 Objects		Th = 7 Objects	
	TPR	FPR	TPR	FPR	TPR	FPR
Navigation	91.49	9.09	93.48	5.88	97.83	2.94
Manipulation	66.67	34.21	73.81	26.32	81.4	16.21

6. CONCLUSION AND FUTURE WORK

In this paper, two processes of plan generation have been developed. The first process is suitable for domains with deterministic world states, while the second process is designed to handle domain uncertainty in world states.

The novelty of the first approach comes from generating semantic plans by using semantic knowledge to compute implicit expectations. These expectations are involved in verifying the generated plans when plan actions are generated successfully. These implicit expectations are then checked based on the evidence information collected by the robot.

In the second process, a new method has been developed to enable the robot to deal with uncertainty. A template of the MLN model is created based on information stored in the semantic knowledge base. This model has the ability to learn by using training data in order to obtain learned MLN, which has the ability to answer queries when generated from the planning system. The learned MLN has the ability to infer (in a probabilistic way) the types and the existence of objects or places in the robot environment. The values of MLN answers can be matched, unmatched or ambiguous depending on the weights distribution of the MLN formulas and the probability associated with each answer.

The experimental results show that semantic domain knowledge can effectively help robots achieve good performance when it is used to generate plans for their tasks.

Table 2. The rates of true positives (TPR) and false positives (FPR)

	Th = 3 objects		Th = 5 objects		Th = 7 objects	
	TPR	FPR	TPR	FPR	TPR	FPR
Navigation	100	17.91	100	15.15	100	10.45
Manipulation	100	80.65	100	67.74	100	59.38

The use of semantic knowledge in plan generation helps detect incorrect cases that cannot be detected by traditional planning approaches because planning by depending only on explicit effects of actions cannot be enough to reveal unexpected situations.

When uncertainty is taken into account, the performance is better as shown by the TPR and FPR results of plan generation of two different types of action. The reason for this increased performance can be attributed to the fact that expectations are not treated in a Boolean manner, i.e., matched, unmatched, or ambiguous. Furthermore, decisions regarding whether the generation of an action has failed or succeeded are based on how likely each expectation is to be verified or violated given the evidence information.

The focus is on generating robot plans by checking its (explicit and implicit) states: these states may include the robot's location in the case of navigation actions, but also include other aspects of the world state for additional actions, like grasping actions.

Future work includes obtaining more training data in order to cover most of the situations that may face the robot operations and using it to learn about the model and recover from situations when the domain parameters are not deterministic. The use of other types of statistical relational models, such as Bayesian Logic Networks in supporting the task planner will also be considered.

REFERENCES

Al-Moadhen, A., Packianather, M., Setchi, R., & Qiu, R. (2014). *Automation in Handling Uncertainty in Semantic-knowledge based Robotic Task-planning by Using Markov Logic Networks* (Vol. 35, pp. 1023–1032). Procedia Computer Science.

Al-Moadhen, A., Qiu, R., Packianather, M., Ji, Z., & Setchi, R. (2013). *Integrating Robot Task Planner with Common-sense Knowledge Base to Improve the Efficiency of Planning* (Vol. 22, pp. 211–220). Procedia Computer Science.

Baader, D. C., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2010). *The Description Logic Handbook: Theory, Implementation and Applications*. Second Edi.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL Web Ontology Language Reference. *W3C*. Retrieved December 15, 2013, from http://www.w3.org/TR/owl-ref/

Bouguerra, A., & Karlsson, L. (2004). Hierarchical task planning under uncertainty. Proceedings of the 3rd Italian Workshop on Planning and Scheduling, 9th National Symposium of Associazione Italiana per l'Intelligenza Artificiale, Perugia, Italy.

Bouguerra, A., Karlsson, L., & Saffiotti, A. (2007). Handling uncertainty in semantic-knowledge based execution monitoring. *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, (pp. 437-443). doi:10.1109/IROS.2007.4399317

Eich, M., Dabrowska, M., & Kirchner, F. (2010). Semantic labeling: Classification of 3d entities based on spatial feature descriptors. *IEEE International Conference on Robotics and Automation (ICRA 2010)*.

Galindo, C., Fernández-Madrigal, J. A., González-Jimenez, J., & Saffiotti, A. (2007). Using semantic information for improving efficiency of robot task planning. *in: ICRA Workshop on Semantic Information in Robotics, in IEEE ICRA 2007*, Rome, Italy.

Galindo, C., Fernández-Madrigal, J. A., González-Jimenez, J., & Saffiotti, A. (2008). Robot task planning using semantic maps. *Robotics and Autonomous Systems*, *56*(11), 955–966. doi:10.1016/j.robot.2008.08.007

Galindo, C., González-Jimenez, J., & Fernández-Madrigal, J. A. (2004). Interactive Task Planning through Multiple Abstraction: Application to Assistant Robotics. *16th European Conference on Artifical Intelligence*. (pp. 1015-1016).

Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernandez-Madrigal, J. A., & Gonzalez, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2278–2283).

Hoffmann, J., & Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, *14*(1), 253–302.

Hois, J., Wünstel, M., Bateman, J. A., & Röfer, T. (2007). Dialog-Based 3D-Image Recognition Using a Domain Ontology. *Lecture Notes in Computer Science*, *4387*, 107–126. doi:10.1007/978-3-540-75666-8_7

Intelligent Autonomous Systems - ProbCog Toolbox. (2013). Retrieved March 21, 2014, from http://ias.in.tum.de/software/probcog

Jain, D. (2011). Knowledge engineering with markov logic networks: A review. *DKB 2011: Proceedings of the Third Workshop on Dynamics of Knowledge and Belief*.

Jain, D., Waldherr, S., & Beetz, M. (2009). *Bayesian logic networks. Technical report, IAS Group, Fakultät für Informatik*. Technische Universität München.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. *Proceedings of the 22Nd International Conference on Machine Learning*. (p. 441-448). doi:10.1145/1102351.1102407

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Tecniques*. MIT Press.

Laskey, K. B. (2008). MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, *172*(2-3), 140–178. doi:10.1016/j.artint.2007.09.006

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, *38*(11), 33–38. doi:10.1145/219717.219745

Lukasiewicz, T. (2008). Expressive probabilistic description logics. *Artificial Intelligence*, *172*(6-7), 852–883. doi:10.1016/j.artint.2007.10.017

Ong, S. C. W., & Hsu, D.Shao Wei Png; Wee Sun Lee. (2010). Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, *29*(8), 1053–1068. doi:10.1177/0278364910369861

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, *12*(3), 441–450. doi:10.1287/moor.12.3.441

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies, In: *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. (pp.458–463).

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1-2), 107–136. doi:10.1007/s10994-006-5833-1

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks, *Proceedings of the 20th National Conference on Artificial Intelligence*, Vol. 2, (pp.868–873).

Tenorth, M., & Beetz, M. (2009). KNOWROB — knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4261–4266). doi:10.1109/IROS.2009.5354602

Tenorth, M. Kunze, L., Jain, D., & Beetz, M. (2010). KNOWROB-MAP - knowledge-linked semantic object maps. In *2010 10th IEEE-RAS International Conference on Humanoid Robots* (pp. 430–435). IEEE.

Theobalt, C., Bos, J., Chapman, T., Espinosa-Romero, A., Fraser, M., Hayes, G., & Reeve, R. et al. (2002). Talking to Godot: dialogue with a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and System:* Vol. 2, (pp. 1338–1343). doi:10.1109/IRDS.2002.1043940