SEMANTIC BASED TASK PLANNING FOR DOMESTIC SERVICE ROBOTS

A thesis submitted to Cardiff University in

candidature for the degree of

Doctor of Philosophy

By

AHMED ABDULHADI AHMED AL-MOADHEN

School of Engineering

Cardiff University

United Kingdom

April 2015

Abstract

Task Planning is developed for an autonomous mobile robot in order to support the robot to accomplish tasks in various degrees of environmental complexity. This environment can be fixed or deterministic (as in a factory), dynamic (as in the human domestic household), or non-deterministic (as in the space exploration). The robot should be provided with a reliable planning system in order to face its major challenge of being certain that its plan to accomplish a task is generated correctly, regardless of the dynamic or uncertain elements of its environment.

This thesis is focused on providing the robot task planner with the ability to generate its plans reliably and detect the failures in generating correct plans. Previous approaches for generating plans depended mainly on action effects (explicit effects) that are encoded in the action model. This means that the action effects should cover most of the characteristics of the newly generated world state. However, this extra information can complicate the action model, especially in the real world.

In this thesis, a semantic knowledge base is proposed to derive and check implicit information about the effects of actions during plan generation. For example, this approach would inform the robot, that it had entered a bedroom because it has recorded at least one bed and zero ovens. When a robot enters a room, the implicit expectations are derived from a semantic knowledge base about that type of room. These expectations should be verified in order to make sure the robot is in the correct room.

The main contributions of this thesis are as follows:

- The concept of using the Semantic Knowledge Base (SKB) to support the robot task planner under deterministic conditions has been defined. A new model of high-level robot actions has been developed, and this model represents the details of robot action as ontology. This model is thus known as the Semantic Action Model (SAM). An algorithm that integrates SKB and SAMs has also been developed. This algorithm creates the “planning domain” in the Planning Domain Definition Language (PDDL) style. This is used as input to the planner to generate the plan for robot tasks. Then, a general purpose planning algorithm has also been defined, which can support planning under deterministic conditions, and is based on using ontology to represent SKB.

- A further contribution relates to the development of a probabilistic approach to deal with uncertainty in semantic knowledge based task planning. This approach shows how uncertainties in action effects and world states are taken into account by the planning system. This contribution also served to resolve situations of confusion in finding an object relevant to the successful generation of an action during task planning. The accuracy related to this type of planning in navigation scenario, on average, is (90.10%).
- An additional contribution is using the planning system to respond to unexpected situations which are caused by lack of information. This contribution is formalised as a general approach that models cases of incomplete information as a planning problem. This approach includes a sequence of steps for modelling and generating a plan of actions to collect the necessary information from the knowledge base to support the robot planner in generating its plan. This results in developing a new type of action which is known as a Semantic Action Model for Information Gathering (SAM_IG). These actions have the ability to access the knowledge base to retrieve the necessary information to support the planning system when it is faced with incomplete information. The information gathering approach is also used to gather the necessary information in order to check the implicit expectations of the generated actions. The correct classification related to this type of planning in navigation scenario, on average, is (92.83 %).
- Another contribution is concerned with solving the problem of missing information, which is using the methods for measuring concept similarity in order to extend the robot world state with new similar objects to the original one in the action model. This results in developing Semantic Realisation and Refreshment Module (SRRM) which has the ability to estimate the similarities between objects and the quality of the alternative plans. The quality of the alternative plans could be similar to the original plan, in average, 92.1%.

The results reported in this thesis have been tested and verified in simulation experiments under the Robot Operating System (ROS) middleware. The performance of the planning system has been evaluated by using the planning time and other known metrics. These results show that using semantic knowledge can lead to high performance and reliability in generating robot plans during its operation.

Acknowledgements

I thank my God ‘Allah’ for his mercy and kindness to support me in all period of my life.

I owe my parents a great gratitude for all their love, prayer and support. This work would not have been possible without the support, appreciation and patience of my wife Zahraa and our children Ali and Abdullah.

I would like to express my gratitude to all those who have helped me, in any way, to successfully complete my PhD thesis.

In particular, I would like to express my sincere thanks to my supervisors, Dr Michael Packianather and Prof Rossi Setchi and, my supervisor for the first two years of my PhD study, Dr Renxi Qiu, for their encouragement and invaluable advice and guidance throughout my study.

Also, I would like to thank my brothers and sisters for their prayer to support me. I would like to thank Dr Ze Ji for his time that he had spent with me to discuss about the research.

I would like to thank “*coursera*”, an educational platform "https://www.coursera.org/", for providing very good courses which I used online for developing my experiences and skills.

Grateful acknowledgement is made to my country “Iraq” for granting me this opportunity to develop my knowledge and experience.

Dedication

This work is entirely dedicated to my Queen ‘Fatima’ the daughter of Prophet of God Mohammed 'May Allah grant peace and honour on him and his family'.

This work is also dedicated to

My Grandfather

My Parents

My Wife

My Children

Declaration and Statements

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ……..… (Ahmed Abdulhadi Ahmed Al-Moadhen) Date ……

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD

Signed ……..… (Ahmed Abdulhadi Ahmed Al-Moadhen) Date ……

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ……..… (Ahmed Abdulhadi Ahmed Al-Moadhen) Date ……

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ……..… (Ahmed Abdulhadi Ahmed Al-Moadhen) Date ……

Table of Contents

List of Figures

List of Tables

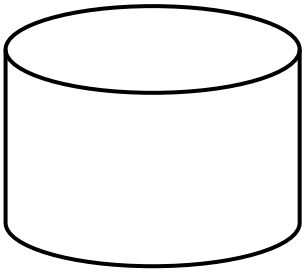
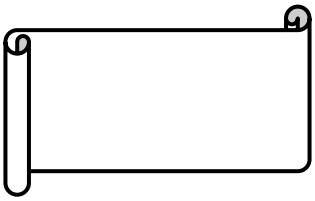
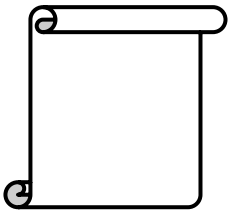
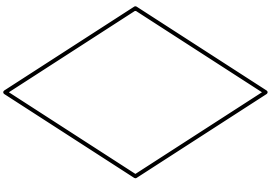
List of Algorithms

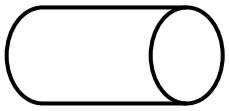
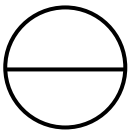
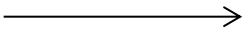
Abbreviations

ABox	Assertion Box
A	Ambiguous Results
BLN	Bayesian Logic Network
DL	Description Logics
FPR	False Positive Rate
GI	Gain in the Information
GPS	General Problem Solver
HTN	Hierarchical Task Network
IG	Information Gathering
IPC	International Planning Competition
Java API	Java Application Programming Interface
KB	Knowledge Base
M	Matched Result
MLN	Markov Logic Network
MN	Markov network
OMRKF	Ontology-based Multi-layered Robot Knowledge Framework

OMICS	Open Mind Indoor Common Sense project
Optop	Opt-based total-order
OWL	Web Ontology Language
POMDP	Partially observable Markov decision process
PDDL	Planning Domain Definition Language
SAM	Semantic Action Model
SAM_IG	Semantic Action Model for Information Gathering
SKB	semantic knowledge base
SRRM	Semantic Refreshment and Realisation Module
SBox	Spatial Box
STRIPS	Stanford Research Institute Problem Solver
TBox	Taxonomy Box
TNR	True Negative Rate
TPR	True Positive Rate
U	Unmatched Results

List of Shapes

	Knowledge Base
	Primary Process
	Input Files
	Output File(s)
	Secondary Process
	Decision

	Data Base (Direct Data)
	Difference
	The direction that the process flows
	Information

Chapter 1

Introduction

1.1 Introduction

Over the years, the development of planning techniques has been one of the main interests in the field of task planning. Task planning is required in several applications such as robotics, web composition, and scheduling.

In various applications, intelligent autonomous mobile robots are being expanded with the purpose of performing complex tasks. These tasks can be local domestic tasks within the homes of humans (Thrun et al. 1999; Saffiotti and Broxvall 2005) at remote planets (Muscettola et al. 1998) and can even be underwater (Fujii and Ura 1996). Robots' environments should be represented in an intelligent way in order to provide these robots with the ability to interact with their environments and perform their allocated tasks. High-level techniques should be employed within robot control architecture to enable a robot to reason about its actions and available resources in an elastic and effective manner.

Modern mobile robots, equipped with artificial intelligence techniques, have the ability and flexibility to carry out a greater range tasks with a minimal human intervention, and without the need to program each task from beginning. Task planning is used to allow robots to generate their own course of action (plan), in order to accomplish a given set of tasks.

Semantic knowledge information plays an important role in representing the structure of a robot's environment, and explains the relationships between the entities (objects and places) and their respective properties.

This thesis aims to provide an intelligent support framework for robot task planning in (real-world) indoor domestic environments. The main focus is on the ability of a mobile robot to semantically generate its own plans and to make sure that these plans are created as expected, based on an improved awareness of the robot's environment. This capability is essential for the performance and autonomy of any moving robot.

Robot plans should not divert from the assigned task. The robot should be provided with the ability to interact with its environment's objects and places, taking into consideration the meaning of these objects and places and their interdependencies.

When unpredictable changes or missing information occurs while an autonomous mobile robot is attempting to achieve its tasks, the robot should have the ability to adapt its behaviour according to these dynamic conditions. This means that the robot should be able to find alternative ways to continue planning its tasks in spite of the occurrence of unexpected situations. This requires alternative methods to support robot task planning.

This thesis presents two general schemes for responding to unexpected situations which impose a lack of information relevant to the task planning. The first scheme relies on information gathering techniques proposed in this study, to extend the robot workspace with new information. The second scheme is based on extending the robot's initial state with equivalent or similar concepts. In the case of uncertain situations, this study proposes a learned probabilistic model, which is obtained by

training the statistical relational models, in order to produce the most probable plan for a given task.

The learned model can support construction of the planning domain by answering queries about uncertain information. This process enforces robot task planning in its duty to generate semantic and most probable plans. This scheme can be applied to infer the most probable information needed to solve uncertainty in situations which relate to robot tasks, either before or while building a plan of actions.

Humans have the ability to create and use symbolic representations of their environment, and this is the main feature that characterises their intelligence (Deacon 1998). These representations enable humans to perform their tasks intelligently, for example, to know what would be the results of their actions before they perform them. Furthermore, humans can learn by testing different types of strategies to find out which is the most useful or superior according to the current conditions of the environment.

The mental representation of the physical environment can greatly simplify the huge amount of information collected by the human senses. The human brain has the ability to abstract sensed information into a general class, i.e. a symbol that represents all objects under this class, regardless of their particular shapes, colours, sizes, etc. This capability allows humans to efficiently process problems of high complexity without considering the huge amount of information arising from their physical environment. They are then able to communicate with each other by dealing with shared symbols.

Nowadays, there are more and more robots that can perform manipulation actions such as picking-and-dropping or even more complex tasks such as setting a dinner table or

cooking meals. These include the Care-o-bot (Reiser et al. 2009), the PR2 (Bohren et al. 2011), TUM-Rosie (Beetz et al. 2009) and Herb (Srinivasa et al. 2009).

These robots will become common-place in the human world once a great development in robot hardware and control routines (software) takes place. The robots need knowledge representation and processing methods to support them in generating suitable plans for their current tasks.

When informal instructions are given by a user to a robot, a lot of knowledge is required in order to understand the instructions correctly. Small differences in instruction details, which may be obvious to a human, would need to be explained to a robot. For example, a human would not describe the opening of a fridge door as part of the process of taking meat from it, or explain that a bottle needs to be opened before the water can be poured into a glass.

Humans can comprehend incomplete information about a task because of their ability to learn and store vast volumes of action results and knowledge, and then quickly retrieve everything they need to know for a given task. Since people can assume other humans have this kind of knowledge, they do not have to explain it when describing how to perform a task or how to deal with an object. The addressee of the instruction should be able to illustrate the information correctly in order to have the utility of reducing redundancy in relationships. Humans can assign meaning to abstract phrases in the tasks (e.g. *fridge*, *isContained* or *grasp*) by relating them to knowledge they already have about these things. For example this knowledge could be that fridges are containers used for storing perishable foods, things are not visible if they are inside a closed container, or the result of grasping an object is that this object is held in the hand.

An important goal in using artificial intelligence techniques in the operations of artificial robotic systems is to endow them with the ability to display expert problem-solving behaviour and reasoning in the real world. Early attempts at realising artificial smart robot capabilities have focused on manipulation of abstract symbols through formal calculi. The central function of mind, thought, which corresponds to the notion of cognitivism, can be accounted for through the rule-based manipulation of symbols (Newell and Simon 1976). Knowledge (or abstract knowledge) is the systematic representation of the world, and it describes the heart of the cognitive system.

Research on embodied cognition (Anderson 2003) has largely avoided symbols, abstract planning and logical calculi. It has instead emphasised the physical implementation of an agent and its interaction with its world. Consequently, research into embodied cognition typically insists on working exclusively with symbols that could be easily related to the cognitive system's particular manifestations of perceptions and actions. Hence these symbols are physically grounded. The physical grounding hypothesis (Brooks 1990) rejects the notion of symbols that assume a knowable objective truth, and states that grounding is at the root of intelligence. The key proposition is that it is better for a robot to "use the world as its own model" (Brooks 1991) rather than using some abstract symbolic representation of it. This view of cognition ultimately calls for a bottom-up, developmental approach, in which abstractions are somehow constructed from grounded low-level perception and experience.

Two separate subfields of Artificial Intelligence (AI) research can be recognised: one has focused on logical representation, and the other on statistical representation. Logical AI approaches tend to emphasise handling complexity. Examples of logical AI

include logic programming, description logics, classical planning, symbolic parsing, and rule induction. Statistical AI approaches tend to emphasise handling uncertainty. These include Bayesian networks, hidden Markov models, Markov decision processes, statistical parsing, and neural networks. Intelligent agents must be compatible with both types of AI approach if they are to be useful in real-world applications.

1.2 Motivation

In real-world environments a range of events can occur which present challenging issues for robots. Situations which lead to confusion in the robot plan generation are a typical cause of such issues. For example, a locked door might render the robot arm unable to open the door and navigate to a goal destination. When a robot generates a plan which involves exploring, detecting and recognising objects, it may find itself incapable of preforming that plan due to unfavourable lighting conditions for capturing clear images of the environment. Another example may be that a robot cannot find “the green cup on the kitchen table” because another robot has picked it up and placed it in the cupboard. A robot may also fall down some stairs because it was unaware that they were there. The list of examples is endless, but the key point is that in all those cases the robot failed to generate a suitable plan which takes into consideration the semantics and conditions of the robot world.

Generally, failures in generating suitable plans are detected when the robot planning system finds itself in situations that it did not expect. The uncertainty, complexity, and dynamics of the environment are common unexpected situations which may occur during a robot's operations. Uncertainty itself can be the result of many factors. The

robot sensors are an important source of uncertainty because they carry an inherent degree of error due to noise and physical limitations. For example lighting conditions must be within certain limitations in order for some optical systems to function. Most failures in a navigation plan are caused by errors in localising the robot within its environment, which is in most cases due to poor sensing. Unreliable sensors provide uncertain measurements which can lead to incorrect robot state estimation (e.g. robot pose) and incorrect control actions. As a result, the robot might fail to generate plans for its future goals.

The work presented in this thesis aims to contribute to the design of autonomous and intelligent robot planning systems which will support the next generation of robots to work closely with humans and assist them in their tasks. It may be many years before researchers discover the mental processes which govern human intelligent behaviour. Moreover, if this feat in some day achieved, it is not clear how such mental processes could be replicated and implemented in intelligent machines (Bickhard and Terveen 1995; Hauser 1997; Nualláin et al. 1997). In the meantime, small but steady steps can be taken to approach the problem.

This thesis will develop and present new ideas, algorithms and mathematical formulas to enable mobile robots to face complex tasks within their domestic environments. To achieve this, the initial challenge is simplified into a robotic application in which a mobile robot semantically plans and carries out its tasks, managing knowledge bases stemmed from a large indoor scenario, whereas it interacts intelligently with humans.

1.3 Scope of the Thesis

Planning approaches for robotic system tasks have generally focused on comparing between the current robot world state, initial state or states resulting from the effects of applying certain actions. These effects are specified in an action model which is primarily used by the planner to extract the explicit effects of actions. The aim of the comparison is to select the action whose its conditions are verified, and then insert this action successfully into the plan. Examples of such approaches include the ROGUE mobile robotic architecture (Haigh and Veloso 1997) and the work of (Fichtner et al. 2003).

If unexpected situations occur, such as uncertainty or incomplete information, the planning system should be supported to generate the most probable plans or to extend its world states with new information.

Creating a robot plan using only the explicit effects of actions on static world states theoretically means that the predicted outcomes are directly notable. However, that is not always realistic in complex environments where extending world states with expectations is an inherently complex process. For this reason the primary focus of this thesis is on using of advanced knowledge representation methods and reasoning techniques involving semantic domain knowledge to derive and infer implicit expectations related to the correct generation of robots' action plans.

Semantic domain knowledge refers to knowledge about objects, places, and their classes, as well as how those objects and places are related to each other. For instance, in a kitchen environment, a kitchen is a class whose individual instances (objects) are rooms that have an oven or cooker. The entities *ovens* and *cookers* denote classes of

kitchen objects. In the context of robot task planning, semantic domain knowledge is used as a source of information to logically derive implicit expectations from the explicit ones. These explicit expectations are the ones encoded in the action models. Semantic domain knowledge is also used to extend world states. The key idea is to compute implicit expectations that can be checked at planning time to make sure that actions are generated as expected. For example, if the mobile robot needs to move into a room that is of type *office*, then the robot will find itself in that room (explicit expectation), and it will observe objects that are typical of an office (implicit expectations), e.g., a desk, a chair, and possibly a PC. If the robot needs to enter a kitchen, it will expect to see an oven, a sink, etc.

Unexpected situations that can occur at the planning stage are also addressed. The assertion of this thesis is to investigate unexpected situations caused primarily by a lack of the information required for planning robot tasks. The focus will be on cases where (a) a lack of information is noted during the planning stage; (b) the outcomes of an action has only partial information about holding implicit expectations; and (c) the robot may not have enough knowledge to find the appropriate object due to several objects sharing some common properties.

The following constraints are used in this thesis when dealing with the identified problem:

1. Plan generation: this work is restricted to dealing only with the formulation of high-level symbolic plans. This means that low-level planning and execution is not addressed in this thesis.

2. Single mobile robot: this work considers the generation of task plans for a single mobile robot. Multi-robot plan generation, although challenging, is not addressed in this work.
3. Indoor domestic environments: the robot task plans are based on an indoor domestic environment. Tasks that involve outdoor environments are not considered.
4. Semantic knowledge base: the semantic knowledge base contains the main object classes in the robot environment and relationships between these classes. It is used in the planning process to provide the planning system with the implicit information of the planning time object.
5. Evidence base: contains the actual objects and their properties in the robot environment during robot operation. The evidence will be used in the planning process to check if the planning time object is matched with an evidence object or not.
6. Planning system: refers to the overall components such as planner, semantic knowledge base, evidence base and all the units that will be addressed in this study.

1.4 Research Methodology

In this thesis, the discipline of artificial intelligence has been used to develop solutions to robot task planning. This has been achieved by generating semantic plans using standard tools and techniques to allow robots to respond to unexpected situations. To address the problem of semantically generation of plans, a semantic knowledge domain is employed as a source of information to compute and check the conditions

that should hold when an action is selected correctly. The notion of the Semantic Action Model (or SAM for short) is defined, and an algorithm for transforming SAM into the Planning Domain Definition Language (PDDL) format is proposed. Then a general planning algorithm which based on the use of description logics (DL) for representing the knowledge is developed. This model will be used in deterministic planning where the knowledge of the initial state is complete and static. A second approach of planning is developed to take into account probabilistic uncertainty, both in action outcomes and world states. In particular, in this type of planning, the planning system is allowed to have action models with more than one possible outcome, and the uncertainty about the state of the world, due to action's outcomes, is also taken into consideration. This extension is essential to the applicability of the proposed approach, since uncertainty is a widespread phenomenon in robotics. The third planning type will be used for incomplete information situations, and in this case the planner is used to gather necessary information to support task planning and to generate the plan. A fourth approach of planning is developed by taking into consideration the handling of missing information by acquiring support from a Semantic Realisation and Refreshment Module (SRRM) then the robot world states are extended with equivalent and similar objects.

Since this study is dealing with simulated problems, the best way to validate the proposed approaches is through carrying out an experimental evaluation. So, an extensive simulation experiments have been performed in order to collect data for the purpose of statistical evaluation of performance. Unfortunately, the lack of shared benchmarks in the field makes the evaluation against other solutions impossible. In fact, a common problem that is faced by research works like this study is how to

evaluate performance. This problem is mainly due to lack of appropriate evaluation metrics, which are available in other research areas.

1.5 Aim and Objectives

The overall aim of this research was to develop a robust robotic task planner. The first proposed stage of the development was to create a robot knowledge base that reflects the robot environment and the entities (objects and places) within it. This knowledge was represented as ontology which consists of the main objects and places classes and their individuals. The second stage was to construct Semantic Action Models (SAMs) that represented the robot action details as ontology in order to facilitate its integration with the knowledge base. In these two stages the planning system depended on the deterministic information which was stored in the knowledge base. The third stage dealt with uncertainty information by using a combination of probability theory and logic theory through Markov Logic Networks (MLNs). The fourth stage also dealt with incomplete information. This stage depended on the planning system to gather the necessary information to support it in generating a plan for its tasks. The fifth stage was to deal with cases when the robot faces situations of missing information related to its current task, but in a different way. This stage depended on the similarity techniques and the calculation of the quality of alternative plans.

The objectives of this work were:

1. To develop a Semantic Knowledge Base (SKB) to represent the robot environment as an ontology.

2. To create a new model of action to represent the details of robot action. This model is called the Semantic Action Model (SAM).

3. To develop an algorithm that integrates SKB and SAM to generate PDDL files which are input to the planner to generate the plans.

4. To propose a general task planning algorithm to check whether the explicit effects of actions are matched (or not) with the implicit expectations of action effects.

5. To deal with uncertainty situations by developing an algorithm to create an MLN model from SKB and then train this model to be ready to answer queries which are generated from the planning system.

6. To develop information gathering technique to support the robot task planner in incomplete information conditions by collecting new information from knowledge base and extending the world state with this information.

7. To develop a Semantic Realisation and Refreshment Module (SRRM) which is responsible for estimating similar objects to the original object in the action model in order to recover the robot planning system from situation of missing information (target). Also, SRRM is used to calculate the quality of the alternative plans which are related to the similar objects in order to select the best one.

1.6 Thesis Statement

This thesis is about using standard artificial intelligence knowledge representation and reasoning techniques to make robot task planning more robust. The thesis statement is:

Semantic domain knowledge and semantic action modelling increase the robustness of autonomous robot task planning because they contribute to the detection and handling of unexpected situations during plan generation.

1.7 Thesis Outline

The work in this thesis can be read in a sequential style. The interested reader may also able to concentrate on individual parts, though a basic understanding of the knowledge representation, reasoning and planning concepts used would also be helpful. Figure 1.1 summarise the outline of the thesis and where the research questions are addressed.

Chapter 2 introduces the main literature which is used to construct the scientific foundation of the work presented in this thesis.

Chapter 3 describes in detail the main tools which are used to create and support the planning system. These tools include knowledge representation, action representation, reasoning techniques and planning fundamentals.

Chapter 4 presents planning in deterministic situations and constructing the semantic knowledge base and semantic action models which are used to produce the planning domain and problem.

Chapter 5 deals with uncertainty situations when the action model has two outcomes, and with how to solve the problem of confusion in finding an object which is relevant to the successful generation of an action plan for a given task.

Chapter 6 describes how the planning system process can be used to gather the necessary information to generate task plans when the planning system is faced with incomplete information.

Chapter 7 presents the way in which the planning system will deal with missing information and testing the quality of the alternative plans.

Chapter 8 presents the conclusions, contributions and suggestions for future work.

1.8 Publications

- ✓ Al-Moadhen A, Qiu R, Packianather M, Ji Z, Setchi R. Integrating Robot Task Planner with Common-sense Knowledge Base to Improve the Efficiency of Planning. Procedia Computer Science 2013; 22: 211–220. **Best paper award**.
- ✓ Ahmed Al-Moadhen, Michael Packianather, Renxi Qiu, Rossi Setchi, Ze Ji. 2015. Improving the Efficiency of Robot Task Planning by Automatically Integrating Its Planner and Common-Sense Knowledge Base. In: Tweedale J. W. et al. eds. Knowledge-Based Information Systems in Practice. Volume 30, pp 185-199.
- ✓ 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, Procedia Computer Science: Vol. 35. (pp. 1023–1032).
- ✓ Al-Moadhen A., Packianather M., Setchi R., and Qiu R. (2015). Supporting Robot Task Planning in Deterministic and Probabilistic Conditions by Using Semantic Knowledge Base, International Journal of Knowledge and Systems Science (IJKSS). **Accepted**.

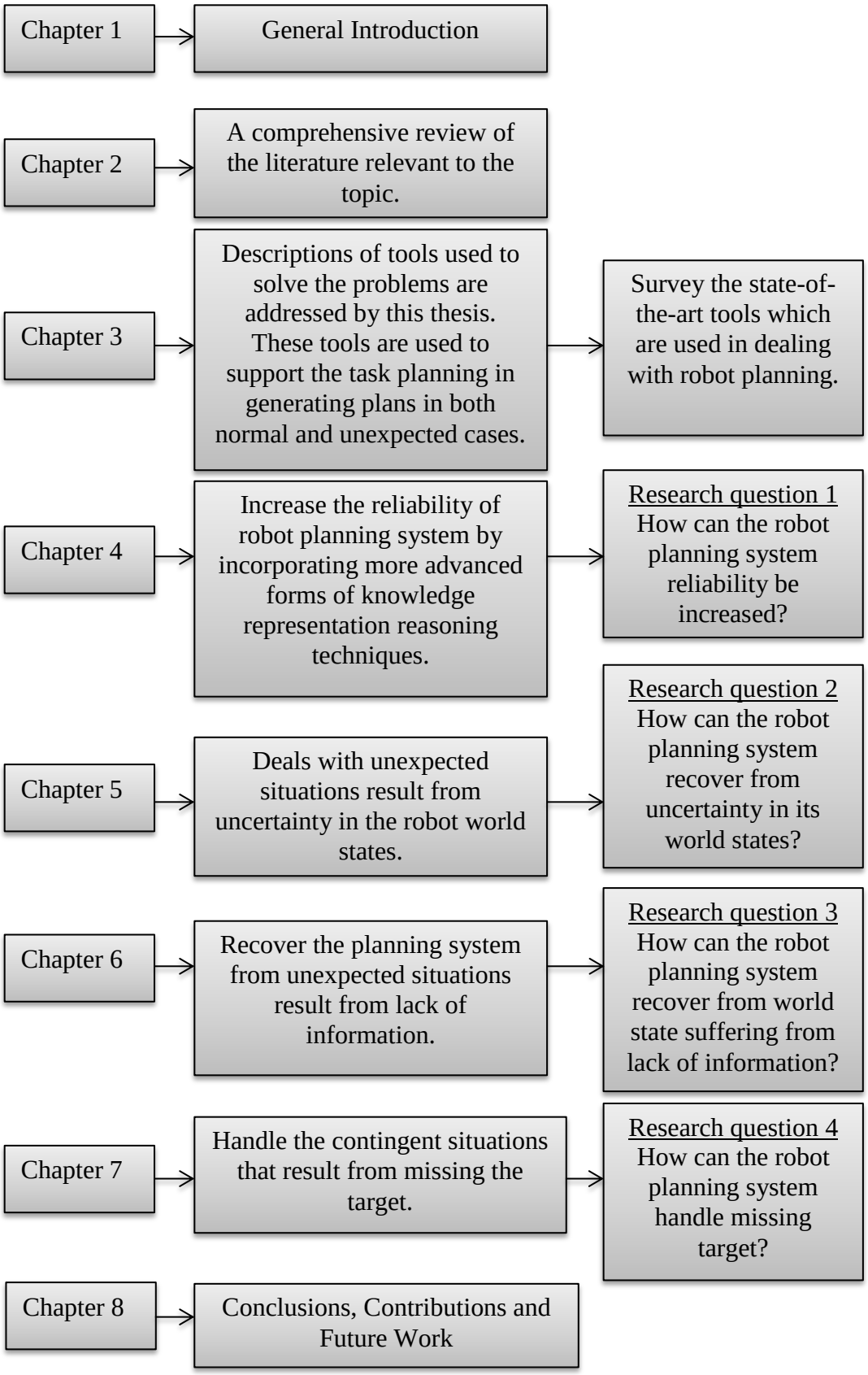


Figure 1.1: A Summary of the Thesis Outline and Research Questions

Chapter 2

Background and Related Work

2.1 Introduction

Plan generation of mobile robots can potentially be a complex and challenging task since it involves dealing with uncertainty in dynamic environments. Autonomy requires that mobile robots are able to recognise unexpected situations that might lead to failures in the plan generation stage for a given task. Autonomy also requires that robots attempt to handle unexpected situations on their own in order to successfully create a plan for their assigned tasks.

Despite the importance of task planning and the need to respond to unexpected situations in the process of plan generation, literature which focusses on these areas is sparse. Instead, such areas are usually mentioned briefly when talking about the general field of task planning.

This chapter will present a review of previous research work in the area of robot task planning and strategies for responding to planning failures. The main focus of this review is the generation of robot symbolic plans.

2.2 Generating of Robot Plans

To schedule their tasks successfully, plan-based mobile robotic architectures need to be able to cope with the issues of uncertainty and the dynamics of the real world that might prevent the correct creation of their task plans. To achieve that objective, planning

systems employ planning techniques and methods in order to make sure that the necessary plan actions are constructed correctly.

The aim of using semantic knowledge in plan generation is to facilitate the planning process and identify unexpected situations that might lead to planning failure. Thus task planning is a fundamental functionality that needs to be implemented in order to achieve robustness in coping with situations that might occur at the planning time. Moreover, the creation of contingency plans is a prerequisite for dealing with unexpected situations.

Most plan generation approaches in mobile robotics use action models to compare the explicit conditions of each action to what is observed in the world states as a result of applying the previous action(s) (Haigh and Veloso 1997; Veloso et al. 1995). It is important to note that the terms ‘symbolic’ and ‘expected’ are also used to describe the situation that might take place when the plan is generated successfully. Conversely, the anomalous situation can be expressed as erroneous, faulty, or simply unexpected.

Figure 2.1 shows the main steps involved in the generation of symbolic plans by mobile robots. Briefly, the task planner takes one action and compares its conditions against the (initial) state of the robot world. If all of the action conditions are verified, the planner will select it. The effect of inserting that action into the robot plan is that the action’s conditions are deleted from the world state and the effects of that action are added to the world state.

The MLN template is created from the SKB and a learning process begins to train the template and generate the learned model. This informed model is then ready to answer

most of the queries which might be generated during the planning process as a result of encountering unexpected situations.

The checking stage compares the estimated state (i.e. that calculated from the evidence database) with the predicted state (i.e., the state resulting from the effect part of the action, the action effect). The aim of this comparison is to check whether there is a contradiction between the two states, indicating an unexpected situation. If a contradiction is detected, a diagnostic process can be launched in order to identify and classify the unexpected situation that has occurred. The main classes of unexpected situations are: (1) uncertainty or (2) lack of information (incomplete information or missing information). The diagnosis result can be used by the planning system to search for a recovery solution or re-planning operation.

Example: Consider a mobile robot trying to plan the task of returning a spoon to a kitchen located in a house and identified by the symbol *k1*. The generated task plan could include the following actions to achieve the task:

move(robot, hall, k1), open(robot, door1),enter(robot, k1), drop-spoon(robot, k1)

This plan includes actions that instruct the robot to move from its current position *hall* to a new position *k1*. Before the robot can enter the kitchen, it should open the kitchen door *d1*, and upon entry should drop the spoon inside the room. Generating the action *open(robot, d1)* implies that the robot has to direct itself until its front camera is facing the door *d1*. Successful implementation of this action within the plan depends on the contents of the robot's evidence database. This must be observed in order to establish the truth value of the predicate *inFrontOf* (*robot, d1*). If the truth value of the predicate is found to be true, then the task planning process deduces that the action has been

successfully inserted in the plan. Otherwise, an unexpected situation is detected, which triggers a recovery procedure with the aim of finding a solution. This solution may be a second plan to achieve the goal of opening door *d1*.

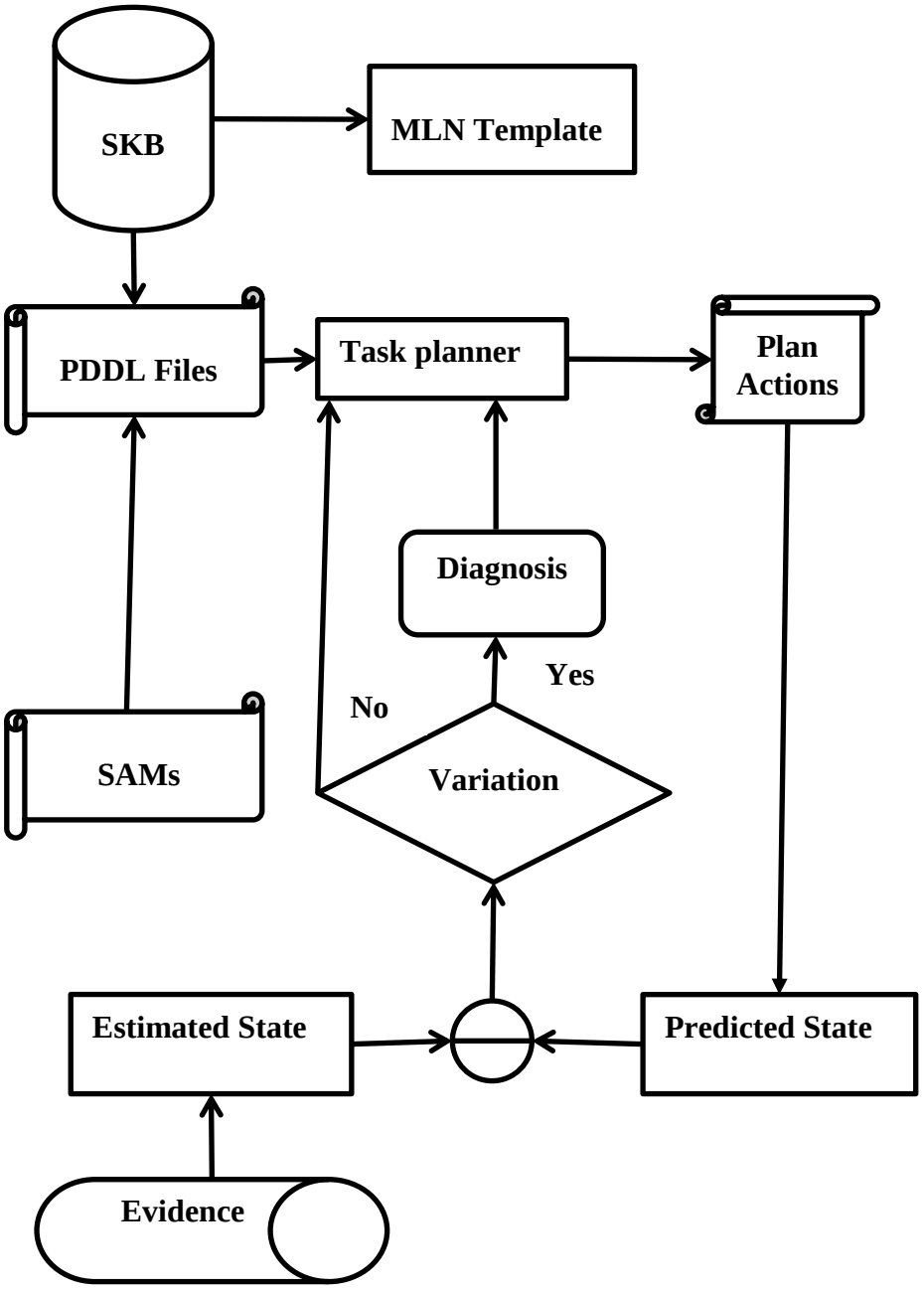


Figure 2.1: Steps of Learning and Generation of Symbolic Plans by a Mobile Robot.

2.3 Overview of Literature Regarding Task Planning

2.3.1 Reviewing Planning under Deterministic Conditions

Using semantic information to support the robot task planner is a topic of great interest in this field. In recent years, the problem of inferring and utilising semantic information in the context of mobile robot operation has gained substantial interest. This is motivated by the observation that mobile robots can benefit from semantic information in various ways, not least that it helps them to more efficiently carry out their tasks. Additionally, it allows robots to reason about their environment, and can be considered a large step towards bridging the gap between perception and action.

Determining the semantic information required to assist a robot to carry out its tasks is a subject of great research interest. This semantic knowledge can be integrated with spatial knowledge to produce semantic maps (Galindo et al. 2008). These maps help a planner to reason and to infer the types of objects from their places. The map aim is to endow the robot with semantic capabilities by relying on a richer multi-ontology approach where two different hierarchies are used: one for spatial knowledge and one for semantic knowledge. The information in the two hierarchies is combined in different ways to provide the robot with more advanced reasoning and planning capabilities. This approach is dependent on asserted information and does not take into consideration the uncertainty that the robot may face.

Recent experience has shown that another benefit of using semantic information is that it allows any violation of facts in the knowledge base to be detected. Robotics applications can benefit from semantic information which carries common-sense facts (Galindo and Saffiotti 2013). For example, if a robot detects that a toothbrush is in the

living room, and the knowledge base suggests the normal place of a toothbrush is in a bathroom, the robot will register that the toothbrush must be returned to the bathroom. The planner must first initiate a goal to recover this violation. The work proposed to use categorical knowledge to achieve goal autonomy. For example, in the categorical knowledge it is known that milk is a perishable substance, and perishable goods should be stored in the fridge. If a robot finds a bottle of milk on the table, it could use this knowledge to generate a task to return the milk to the fridge.

In recent work (Blodow et al. 2011), semantic information was used to build a robot map. The building process took into consideration the semantics of the objects and places in the robot's environment. Semantic mapping of a kitchen environment enabled the robot to perform manipulation tasks. The framework presented in the work makes a clear distinction between knowledge and reasoning by separating them into two entities.

Galindo et al. (2004) improved the task planner efficiency through abstracting robot world definitions. The improvement was achieved through optimised computational efficiency. This was enabled by beginning the planning process at a higher level, and then stepping down and refining the plan by discarding objects or places irrelevant to the current task.

The semantic information discussed by Galindo et al. (2005) could be acquired by a robot's sensor, and then used to build semantic maps. These maps could be organised in a multi-hierarchical fashion and used to support the robot when navigating its environment. The robot uses this semantic level for reasoning, e.g. "this room contains no sink; therefore it cannot be the kitchen".

The linking process between spatial and semantic information is carried out by using of the anchoring technique (Coradeschi and Saffiotti 2003). Object anchoring (or symbol grounding) is an area of robotics-related research that aims to build and maintain new and existing links between symbolic representations of objects (as in a logic-based knowledge representation formalism) and their images in the sensor data stream. In this manner it also relates to semantic mapping. This framework gives the robot the ability to use and infer new semantic information from its environment, leading to improvements in its operation.

Galindo et al. (2007) explored the way that semantic information makes complex robot task planning more efficient. This was seen as a new benefit of semantic information. Semantic information can be used as a tool to improve the task planning in complex scenarios where other planners may easily find themselves in intractable situations. This leads to robots dealing with a great number of objects, places and actions. Galindo et al. (2007) built a semantic plan which classified objects, places and actions, and then presented a generalised version of the requested task. Afterwards, the plan could be used for removing redundant information to enforce the robot planning in an efficient manner.

Other research has focused on semantic knowledge as a source of the information necessary to monitor plan execution. Bouguerra et al. (2007) presented an intelligent plan-execution monitoring method, which used semantic information in the robot domain to extract implied expectations. The robot could then use these expectations to verify the correctness of executing actions within its plan.

A new knowledge processing system, known as KNOWROB, was presented by (Tenorth and Beetz 2009), and was designed for enabling personal robots to be more

autonomous in their work. This system contained the necessary control routines and action sequences to automatically execute a particular requested task. KNOWROB integrates encyclopaedic knowledge with an environmental model, action-based reasoning and human observations. It has the ability to access all this information in a uniform and symbolic way. The map which was used as a source of information by the KNOWROB knowledge processing system is called KNOWROB-MAP (Tenorth et al., 2010). This map includes spatial knowledge about objects in the environment and their relationships, and provides this object details to a robot.

Semantic information can be used with spatial categorisation to build a model which can be used to enable the robot to engage in high level communication with humans (Mozos et al. 2007). This model is composed of layers that represent maps (such as metric, navigation, topological and conceptual maps) at different levels of abstraction.

Tenorth et al. (2010) listed some challenges that can hinder robot operation in unstructured environments, and then proposed methods to overcome them. For example, household robots work in real-world environments that can be highly dynamic, so the robots need specific methods to deal with these uncertain situations. The authors' method for dealing with these situations was to create probabilistic first order models by using statistical relational models to represent probabilistic knowledge.

A good action representation that integrates a good planning algorithm is important for the task performance of many intelligent systems. Kemke and Walker (2006) focused on creating integration between a STRIPS-style (Stanford Research Institute Problem Solver) action hierarchy and a plan decomposition hierarchy. This integration seems ideal for natural language processing as it allows the robot to be more expressive and efficient.

A framework for representing the knowledge of household service robots was proposed by Lim et al. (2007). This framework was divided into classes, axioms and rules to enable a robot to recognise objects and navigate while inferring localisation-related knowledge. This could be performed even if there was hidden and partial data due to noise in the sensed data. Lim et al. (2007) constructed a robot knowledge framework based on Robot-centered ontologies, Human-centered ontologies and various rules. This framework is known as OMRKF (Ontology-based Multi-layered Robot Knowledge Framework). Lim et al. (2007) also conducted a cup delivery service experiment to demonstrate the validity of the framework.

The task planner presented by Galindo et al. (2007) was improved by optimising the symbolic representation of the robot environment using a genetic algorithm. A framework was proposed allowed an indoor mobile robot to automatically construct a symbolic model of its environment. This model is optimised over time with respect to alterations in the environment and the robot operational needs through an evolutionary algorithm. To efficiently process the large amounts of information that the real world provides, an abstraction was used. This also improves the efficiency of task planning process.

Tenorth and Beetz (2013b) provide an overview of several different types of knowledge, along with inference mechanisms for acquiring the knowledge from external sources. A description is then presented of the overall system architecture which includes the main components of (i) knowledge acquisition, (ii) automated reasoning, (iii) visualisation and (iv) information querying.

New representations of environment maps presented by Tenorth & Beetz (2013), combined expressive semantic environment models with techniques for selecting

suitable maps from the web-based knowledge base. The planning domains can be generated by integrating semantic action models with a common-sense knowledge base (Al-Moadhen et al. 2013). These domains are input into a robot planner to generate a suitable action plan for a given task.

Further work in this area (Galindo et al. 2004) included developing a formalism of a symbolic model of the environment to solve the issues associated with processing large amounts of information during planning. This model was efficient in providing human–machine communication in a normal form through a mechanism inspired by humans whereby knowledge is structured 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 is dependent on deterministic information.

2.3.2 Reviewing Planning under Probabilistic Conditions

Planning under deterministic conditions presents few issues compared to planning under probabilistic conditions. Approaches in the previous section have focused on planning under deterministic world states and action details (preconditions and effects). In general, predefined action models are used to describe the states of the environment after actions have been correctly executed.

Bouguerra and Karlsson (2004) proposed a hierarchical task planning approach that handled uncertainty in both the state of the world and the effect of a given action. It introduced mechanisms to handle situations with incomplete and uncertain information about the state of the environment by using belief states to represent incomplete information about the state of the world. Meanwhile, 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 and Tsitsiklis 1987) provide an essential mathematical framework for the planning of autonomous robots in uncertain and dynamic environments. Solving POMDPs exactly is computationally expensive and intractable. For this reason, (Ong et al. 2010) used a factored model to separately represent the fully and partially observable components of a robot's state. They then derived a compact lower-dimensional representation of the robot belief space. This factored representation can be combined with any point-based algorithm to compute approximate POMDP solutions, and hence it improves the speed of POMDP planning under uncertainty.

To deal with uncertainty, probabilistic methods can be used to link the system variables to one other and construct a network among them. Markov Logic Networks (Richardson and Domingos 2006) and Bayesian Logic Networks (Jain et al. 2009) can be utilised to process probabilistic information for supporting many applications including task planning.

Plan generation can take into account the uncertainty in the existence of objects, given their types and properties, by proposing a new framework to construct plans based on probabilistic values which are derived from Markov Logic Networks (MLN). This approach has also been adopted in more recent work (Al-Moadhen et al. 2014). In this framework an MLN module is established for probabilistic learning and inferencing. This is combined with semantic information to provide a basis for plausible learning and reasoning services in the support of robot task planning.

The gap between the semantic and spatial representations of an environment was bridged by (Eich and Goldhoorn 2010). This was achieved by depending on the semantic side, which comes from (Galindo et al. 2008) and (Galindo et al. 2005), to describe the way in which semantic maps can be used for high level planning and spatial reasoning. (Eich and Goldhoorn 2010) described the bridging between the spatial domain and the semantic domain which they refer to as the SBox (spatial box) and the TBox (taxonomy box) respectively. The semantic interpretation of physical objects is achieved by means of optical marker identification, although it is not based directly on the spatial interpretation of point cloud data.

2.4 Techniques for Planning System

Since the research of planning systems has been an active field for at least the last 40 years, several different techniques, algorithms and systems have been proposed. They differ substantially either in the way that they model the world or in their manner of searching for a solution to a problem. They also differ in the way that they employ heuristic techniques to accelerate the recovery process.

The first planning system that appeared in the literature was known as the General Problem Solver (GPS). It was developed by (Newell and Simon 1963), who attempted to simulate human thinking in order to solve problems. A different system proposed in combination with the Stanford Research Institute Problem Solver (STRIPS) formalism, and with the same name (STRIPS), attempted to use state-space planning (Fikes and Nilsson 1971). This class of planning was constantly transforming states by applying existing action details and the heuristic in order to obtain a desired state and reduce the

number of conditions examined. The starting point of the search can be either the initial state or the final state, and hence the planning can be characterised as forward or backward respectively.

Another technique discussed during the 1990s was plan-space planning. This technique also attempted to solve the problem of planning robot actions. The three main examples of plan-space planning are: UCPOP systems (Penberthy and Weld 1992), TWEAK (Chapman 1987) and SNLP (Mcallester and Rosenblitt 1991; Knoblock and Yang 1993). In these systems a search begins with an initial plan which is continuously transformed until the system finds a plan whose accuracy and applicability are ensured. The design of the plan-space search significantly reduces the search time, and offers the possibility of building partially ordered plans.

One of the initial approaches to action planning is a planning hierarchy, in which the user can define different levels of abstraction to describe the problem. Solutions begin at the highest abstract level and the plan is gradually enriched with details from the lower levels so that the conditions are simple enough to be matched with the available action preconditions. In this way it is possible to reduce the search time, but this process requires a prior knowledge of the domain from the domain expert in order to achieve a proper decomposition of actions. Examples of systems that used a hierarchical approach are ABSTRIPS (Sacerdott 1974) and AbTWEAK (Yang and Tenenberg 1990).

The approaches mentioned so far belong to the group of classical action planning techniques but are still currently used in plan construction for planning systems. However, since the early 1990s researchers have turned to new methods to accelerate the planning process and handle issues not addressed by the majority of classical

techniques, such as time, uncertainty, the availability of restricted resources (Vrakas et al. 1999).

A typical method for representing different types of planning techniques is to use graphs. This is known as graph-based planning. The main system that has implemented this technique, and has received special interest in recent years, was Graphplan (Blum and Furst 1997). A key role of using graphs in the planning process is to provide a visual description of the problem which contains all the available information. The graph can then be used to search for reasonable results even for nonlinear plans (conditional plans). Other examples of graph-based planning systems include STAN (Long and Fox 1999) and IPP (Koehler et al. 1997).

An alternative system which depends on graphs to formalise its action planning is the LPG-td (Gerevini et al. 2004; Gerevini and Long 2005). This system has been used extensively in the present study as an external action planning system.

Another important approach used automatic heuristic mechanisms to extract the knowledge from the structure of a problem and to direct the search for a solution. One of the first systems related to this approach was UNPOP (Mcdermott 1996; McDermott 1999). This was followed by the ASP / HSP (Bonet et al. 1997; Bonet and Geffner 1999), FF (Hoffmann 2000; Hoffmann and Nebel 2001) and HAPRC (Vrakas et al. 2005) systems.

A significant number of approaches have attempted to address the action planning problem by transforming it into a different type of problem, such as a problem of Satisfaction Propositional Logic (Propositional Satisfiability). Examples of systems which have use this approach include SATPLAN (Kautz and Selman 1992; Kautz and

Selman 1996) and BLACKBOX (Kautz and Selman 1998). Other researchers have attempted to convert the problem into a problem of Constraint Satisfaction (Constraint Satisfaction Problem), solving it with constraint satisfaction methods and then converting the solution into an action plan. Examples of such systems include IxTeT (Laborie and Ghallab 1995) and HSTS (Muscettola 1994). Finally, planning problems can be represented as model control problems (Model Checking) and Markov Decision Processes. These are techniques which can handle uncertainty (Giunchiglia and Traverso 2000).

Table 2.1 summarises the aforementioned planning techniques with a brief review and examples of systems that represent each technique. The decision to use LPG-td as the main action planning system for solving planning problems in this thesis was based on the following reasons:

1. It is one of the most modern design techniques with a proven performance record in action planning competitions for various kinds of domains and problems. Its performance was verified by experiments in this thesis, as it will be seen in later chapters.
2. It has the ability to handle large numbers of actions in a domain, and a large number of objects in the problems, while minimising the time required for planning. Its status is particularly important for scaling up the problem of the planning when the domains have a large number of actions. Other planning systems, which is used in this thesis, such as the automatic heuristic extraction based Metric-FF system, showed good performance with a moderate number of actions, but an inability to handle large numbers of actions.

3. It gives increased customisation capabilities and returns several plans. It also has the potential to improve the solutions.
4. Its integration with other existing systems is simple because of its compatibility with the PDDL language.

Metric-FF shares with LPG-td in points 1, 3 and 4 but it cannot handle large number of actions (point 2).

Table 2.1: Summary of Reviewing Actions Planning Systems.

	Technique	**Feature**	**System Example**
Classical Planning System	State-space planning	Transforms states by actions to obtain the desired final state.	STRIPS
	Plan-based Planning	Transforms plans until it finds a sound and workable plan.	UCPOP, TWEAK, SNLP
	Hierarchical planning	Solves the problem at the highest level of abstraction then gradually enriches the plan at lower levels.	ABSTRIPS, AbTWEAK
Other Planning System	Graph-based planning	Constructs graphs from a problem description and then uses them in the search for a solution.	Graphplan, STAN, IPP, LPG-td
	Automatic heuristic extraction	Knowledge extraction from the structure of a problem for using in searching for a solution	UNPOP, ASP / HSP, FF, PGRT, CL, HybridAcE, ChAPrk
	Transformation into another problem type	Convert to problem Satisfaction propositional logic (Propositional satisfiability) or Constraint Satisfaction and then find a solution.	SATPLAN, BLACKBOX, IxTeT, HSTS

2.5 Automated Plan Construction

Planning processes in factories or industrial environments face the problem of planning within a static world. In reality, the world of the robot is not always static. Robot environments are open medium, each user having the freedom to add new places and objects or to modify the existing ones. In order to manage this large and constantly changing world and action volume, a semantic knowledge base can help to automate the task planning.

Despite efforts to develop common standards for describing simple and complex actions, the process of editing and constructing task planning has a high complexity. This is due to the following reasons:

1. The number of actions is increasing due to the extremely rapid development of new robots and their functionalities expanding to cover most services. As a result, the planning process must search through extensive and every growing action domains.
2. Actions may be modified several times after their creation, so the planning process must be able to detect such changes and act accordingly in real time. Many times, although the solution is simple, the planner may carry out the process from the beginning.
3. Various actions (or robots) have been developed by different organisations who have used different concept models and natural language to describe them (or their capabilities). Existing standards (action representations) require communication with the parameter and maintain editorial control, but the semantics of these parameters play a key role in the plan creation. The

identification and mapping of world facts and concepts with the parameters is a time consuming and difficult process.

It is logical that increasing robot complexity will increase the time required for a non-automated planning process. Therefore, the creation of an action plan should be carried out using techniques which favour automation. These techniques generally fall in the area of Artificial Intelligence and facilitate the automation of the entire process of task planning, the selection of appropriate actions for individual tasks and the production of a sequence of actions as a final synthetic plan. However, the application of most of these techniques is not possible without semantic information. The use of semantic information makes the planning process more accurate and efficient.

The remainder of this section presents techniques which have been proposed for automating task planning along with illustrative systems that have implemented the appropriate technologies.

2.5.1 Planning in Hierarchical Networks

A Hierarchical Task Network (HTN) is an action planning technique that solves a problem at various levels of abstraction based on hierarchical decomposition of network processes (Erol et al. 1994). An initial network of processes (task network) is entered into a planning system, which also contains the problem to be solved, i.e. the goal of the task planning.

A task network consists of abstract tasks defined by other tasks. There are three types of tasks: (i) primitive tasks which correspond to simple STRIPS (Stanford Research

Institute Planning System) operators and can be run directly, (ii) compound tasks which are composed of a set of simpler tasks, and (iii) goal tasks that are described by conditions representing the desired properties of the final state. Moreover, a set of operators should be given, which describe the results of each primary task (each individual action). A set of methods that determine how complex tasks can be decomposed into simple tasks should also be given. The planning process initially decomposes successively all the complex processes included in the initial network with the primary concern of maintaining the restrictions. In the second phase, when the network contains only primary tasks, the planning process finds a plan that satisfies all the constraints. The plan is a description of the robot actions and consists only of primary tasks, which are assigned to individual actions.

2.5.2 Planning Actions with Regression Assessment

In these systems, in the general case, the outputs of the desired task represented as a target state. The approach aims to transform an initial state into the target state by using the initial conditions and available robot actions.

The state space is composed of all possible situations that can be obtained by applying an arbitrary number of actions to the initial state. Estimated regression planners use a backward search. Forward searching is a less difficult way to find a solution in the state space. The search is guided by a heuristic estimator, which is determined by using backward chaining. This method links to a problem in a space resulting from relaxation, and ignores the interactions between operators that achieve objectives and deletions.

Thus, the space created is somewhat smaller than the state space, and a planning system can fully represent the means of a regression graph. In order to use this system in the field of robot planning, an extension to the conventional notation of PDDL should be added.

The action execution may result in the production of information, and this is sometimes represented by the effect structure of PDDL. This approach assumes such information as the cost (value) of an operator, which can act as an input to subsequent steps in the plan. The definition of the PDDL language extends to include such values in the definition of the operators (McDermott 2002), so that they have the ability to express these types of data. The new language created after these changes was named Opt (McDermott 2005). Opt was used to amend the existing actions planner Unpop (Mcdermott 1996; McDermott 1999) resulting in the new name Optop (Opt-based total-order). Optop is able to handle the new demands of the language used for task planning problems. The system implements the above assumption experimentally for simple planning problems which represent the problem directly in PDDL. However, this approach does not address issues such as scaling, conditional plans or repetitive plans, and research in this area is still in its early stages (Chan et al. 2007).

2.5.3 Proposed Planning System

The proposed planning system in this thesis uses semantic action models (SAMs) to represent the robot action models. These models are then integrated with a robot semantic knowledge base, which is implemented as ontology to generate the planning domain that consists of actions, predicates and problems. The solution to the problem is

obtained by calling an external planner (Metric-FF or LPG-td). In this study, Metric-FF and LPG-td planners are chosen to generate the plans. This approach can also be used to solve the problem by expanding the Metric-FF or LPG-td planning domain with new information in order to generate new plans, or to re-plan when necessary. To configure the planning problem, the system uses the semantic descriptions of SAMs together with the corresponding ontologies to generate the domain of the plan. Therefore, it does not fully exploit the semantic information contained in them, since the use of ontology does not lead to the conscious semantics of concepts, only to the facilitation of the configuration of the domain. Subsequently, the planning system can invoke semantic enhancement during the action planning process to obtain new plans. In this case, the new generated plans' quality is assessed in order to choose the most suitable one. The planning system can also be used in the case of incomplete information in the robot world states by generating a plan that has a sequence of actions which have the capability to access the knowledge base and provide the world states with additional information.

This system appears to be a promising approach in the field of automated task planning. The details of this planning system will be explained in Chapter 4.

2.6 Summary

This section has reviewed the theoretical basis and process of using the semantic knowledge base (SKB) in supporting the robotic system in its duties and the issues related to planning under uncertainty. The review has highlighted the following points:

1. The SKB has been integrated with spatial knowledge to produce semantic maps that are used to help a robot planner to reason and infer the types of objects from their places.
2. The SKB has been used to monitor the robot plan execution, which used the semantic information in the domain to extract implied expectations. Then the robot can use these expectations to verify the correctness of executing action within its plan.
3. The SKB has been used to build the knowledge processing system in order to enable personal robots to be more autonomous in their works.
4. The SKB has been used with spatial categorisation to build a model which can be used to enable the robot to engage in high level communication with human.
5. A hierarchical task planning approach has been used to handle uncertainty in action effects, where the actions are allowed to have more than one possible outcome.
6. Probabilistic methods can be used to link the system variables and construct a network among these variables. The network of variables reflects the dependencies between these variables and can be formalized as a model to support the planning system.
7. Different techniques and algorithms have been proposed to develop the planning system. The main different points are: in the way they model the world, in the manner of searching the solutions or the way they employ heuristic techniques to accelerate the recovery process.
8. The planning systems would begin searching for the solutions from the initial plan which is continuously transformed until the system finds a plan whose accuracy and applicability are ensured.

9. The hierarchical task network is a planning technique that solves a problem at various level of abstraction based on hierarchical decomposition of network processes.
10. The proposed planning system in this study is using the semantic knowledge base and semantic action model to support its planner (Metric-FF or LPG-td) in order to obtain more reliable planning system.

Having established the mechanisms of using the semantic knowledge base, next chapter will explain the planning domain representation and the knowledge representation methods that are used in robotic applications.

Chapter 3

Planning System Tools

3.1 Introduction

Planning operation is the process of selecting a sequence of actions which, when executed by an agent, will achieve a set of objectives (goals). Planning has been an important active area of Artificial Intelligence research for over four decades. It enables greater autonomy and flexibility in systems and intelligent agents, especially when they operate in dynamic environments. Planning systems have a wide range of real world applications, in field from robotics, logistics and construction, to space exploration, navigation and strategy.

The planning of actions to address problems of high complexity and large search spaces cannot be achieved by simple search algorithms as their solutions are insufficient for complex problems. Alternative techniques of problem solving within the context of action planning have been developed. These include non-linear planning, graph-based planning and hierarchical planning. Systems which implement planning algorithms and techniques are known as ‘planning systems’ or more simply as ‘planners’.

In the general case, an action planning problem can be represented by the initial state of the world, the final (goal) state, and the abilities of the robot, which define its set of allowable actions. These actions are used by the robot to transfer from an existing state to a new one. The result generated by the planner for a given problem is called a plan. The plan is a sequence of fully or partially ordered actions, which can be applied to the initial state in order to produce the desired end state.

All the knowledge and techniques required for a robot to accomplish its tasks should be provided by a knowledge representation and processing system. This system needs to make informed decisions, parameterise robot actions, and plan under complete, incomplete or ambiguous conditions. Thus the system requires descriptions of different component of actions that the robot can perform, the features of objects it can interact with, details of the environment it operates in, and knowledge of the abilities of the robot itself. All these requirements need to be represented in the proposed robot planning system so that all robots can understand their meaning, merge them, perform reasoning and draw conclusions.

This chapter describes the tools which are used to solve the problems addressed by this thesis. These tools are used to support the robot task planning system in generating task plans in both normal and unexpected cases. The normal cases represent situations of deterministic world states and action effects. Unexpected cases represent situations of missing, incomplete in robot world states or probabilistic information in world states and action effects.

Firstly, this chapter gives an overview of the planning system, its domain, its problem style and its definition language. Secondly, the knowledge representation method is described, including the method used to structure this knowledge. Description logics are used to represent the knowledge as a semantic knowledge domain. This representation benefits from the use of reasoning to obtain implicit information from explicit information. The ontology is used to structure this knowledge and linked between its classes by using properties and relationships.

Next, graphical models are used to represent the probabilistic knowledge in the robot environment. Markov network (MN) and Markov Logic Network (MLN) models are

used in this study. An overview is given of the proposed planning system architecture in the context of its different interrelated components in order to better understand their roles in the planning system.

Finally, a brief description of the planning under both deterministic and probabilistic conditions will be presented.

3.2 Planning Domain and Problem Representation

The terms ‘planning domain’ and ‘problem representation’ refer to the systematic coding of all available information about the robot world that is relevant for all the robot’s tasks. The resulting domain represents, in a formal way, the elements of the world which will be used to describe problems. This formalism of knowledge representation plays a key role in the types of problems that can be represented and solved, as each style has a given degree of expressiveness and complexity. There are a large number of languages and standards for describing problems, for example propositional logic or first-order predicate logic (Hamilton 1978).

It is obvious that the more expressive a language is, the larger the range of problems it can represent. Therefore, the representation of complex systems and concepts of the real world is better facilitated by expressive languages. However, the time needed to solve a problem normally increases with the complexity of the language, because it increases the computational complexity of the algorithms involved in the solving process (Bylander 1994).

The next subsections describe the most dominant formalism used for representing planning domains and problems. The elements of this formalism are used to represent the process of robot task planning as operation of problems planning.

3.2.1 The STRIPS Model

The STRIPS (Stanford Research Institute Planning System) model is the basis for representing domains and problems in most conventional planning systems (Fikes and Nilsson 1971), due to its simplicity and naivety. A planning problem in the STRIPS style is represented as a triplet <*I, A, G*>, where *I* is the initial state, *A* is a set of available actions, and *G* is a set of objectives (goals). The states are represented as a set of predicates, based on first-order predicate logic. All elements of the original state of the world that have some relevance to the problem must be explicitly declared in the initial state *I*. The initial state *I* can contain both static and dynamic information. For example, in one particular situation *I* may state that the objects *X* and *Y* are boxes, which is an example of static information as it is assumed to remain unchanged during the action planning. At the same time, the state *I* may indicate that the box *X* is initially on the box *Y*, which is an example of dynamic information, as it is expected to undergo changes during action planning.

The state *G*, on the other hand, is not complete. *I* is not necessary to specify in *G* the final state of all objects of the problem, either because it is implied by the context or because the final state of some objects does not matter to the problem. Consider an example from the field of logistics (Mcdermott 2000), in which there are packages to be transferred between locations using different means of transport. The final location of

the transport vehicles is usually omitted from the definition of the problem because the main object is only the transfer of packages to the desired destinations. It follows that there are usually several solutions that meet the objectives of a problem, so it is more correct to say that *G* represents a set of statements instead of a unique situation. The set *A* contains all the actions that can be used to change the world situations. Any action A_i has three sets of facts which contain:

- Conditions of A_i, i.e. facts that should be verified in the current state of the world to allow the application of action (denoted as precondition (A_i)).
- Facts which are added to the state of the world after the application of an action (indicated as add (A_i)).
- Facts which are removed from the state of the world after the application of an action (denoted as del (A_i)).

Typically, the following rules are applied to the world states formalised by STRIPS:

- An action A_i can be applied in state S if precondition (A_i) $\subseteq S$.
- If the action A_i is applied in *S*, the new state *S'* is calculated as: $S' = [S \cup \text{add}(A_i)] - \text{del}(A_i)$.
- The solution to a problem, i.e. the action plan, is a sequence of actions which, if applied sequentially to the initial state *I*, lead to a situation *S'* such that $S' \supseteq G$.

The operators have variables that can be identified with the available objects, and thus facilitate the encoding of the domain. This significantly reduces the number of actions required to accomplish the task.

3.2.2 Definition of the PDDL Language

Planning Domain Definition Language (PDDL) was originally designed to provide a standard means of coding planning domains and problems. This enabled standardisation of the action planning system, to take part in international competition programs such as the IPC (Bacchus 2001; IPC 2004). However, in subsequent editions, this standard has been enriched and expanded in many ways. Nowadays, it has become a standard in the planning domain and problem modelling community. The PDDL has established itself as the most widely accepted common language for the exchange of information among researchers about planning domains and problems.

The typical description of a PDDL domain structure is shown in Figure 3.1, while Figure 3.2 shows the structure of a typical problem expressed in PDDL style. The following paragraphs give a brief description of the components which form the planning domains and problems.

The first edition of PDDL (Ghallab et al. 1998) mainly reflected the physical properties of the domains for each action planning problem, such as the available predicates and the available operators, without giving special attention to the time or other characteristics. However, the first edition was improved over the years and extended to form a new version. The elements of the PDDL style are divided into subsets referred to as requirements. These facilitate the handling of the planning system domains.

Each domain indicates which requirements it uses, so the planning system is aware of its compatibility with them. In this way, the planning system is able to determine without additional information if it has the ability to manage the given domain. The planning system can simply ignore all the definitions related to the requirements which

it is not able to manage. It should be noted that the valid assumption of domains expressed in PDDL is a closed world assumption. This means that anything not specifically mentioned as true is deemed to be false, unless the definition of the field is clearly stated otherwise.

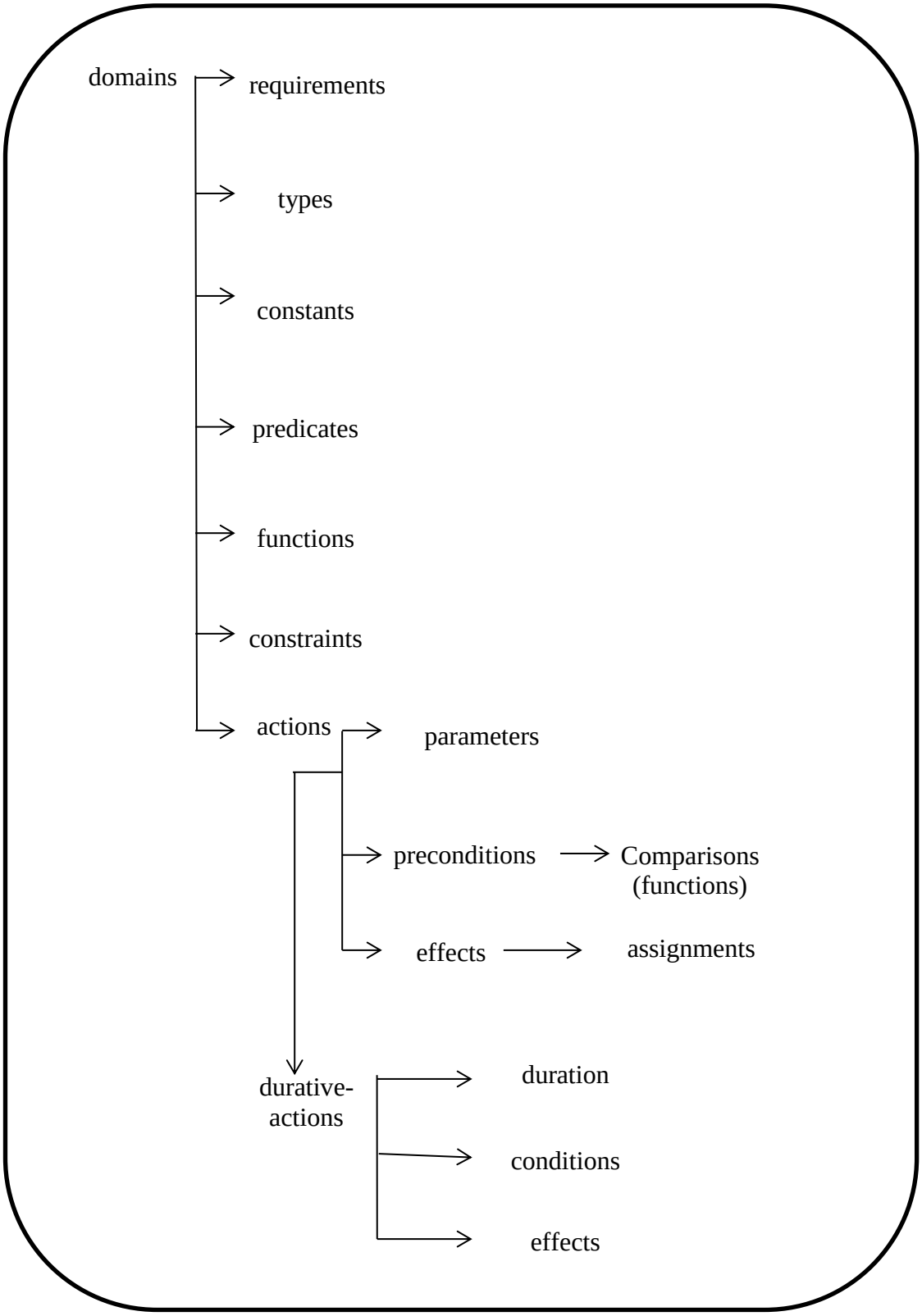


Figure 3.1: The Typical Structure of a PDDL Domain.

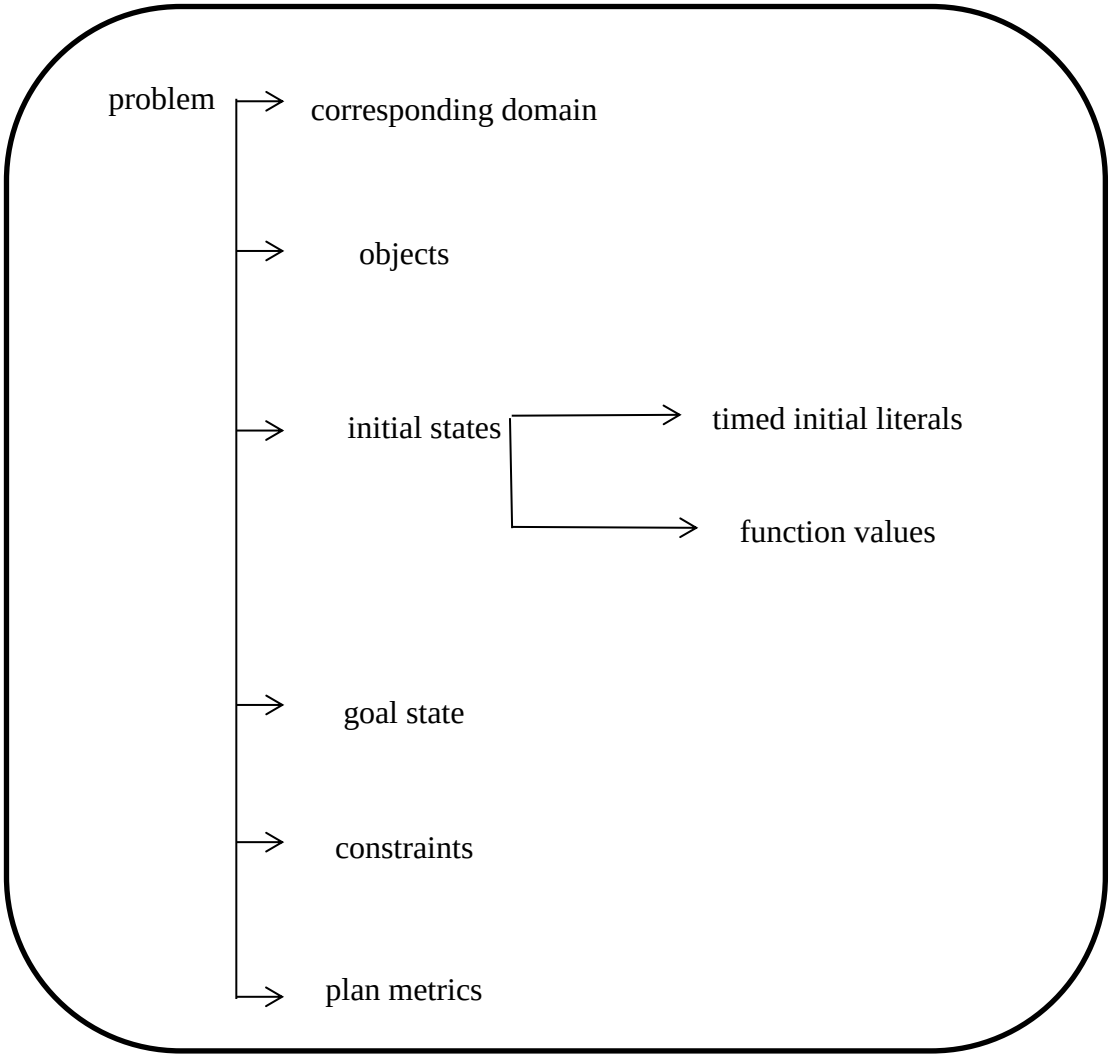


Figure 3.2: The Typical Structure of a PDDL Problem

The using of the term ‘types’ in domain definition is helpful and informative. Domains where the related requirements are stated as (: types), should display every object and predicate argument with their type. If the explicit declaration of the type is absent, then the objects, predicate arguments or variables are considered to be of general type. The definition of types provides a validation feature based on the agreement between object types and argument types by identification. This process is particularly useful for the validation of the domain, i.e. the confirmation of the correctness and consistency.

The variables in PDDL have the same semantics as can be found in many other languages. These can take any value from a set and can be used in conjunction with built-in functions of PDDL for evaluating expressions. In later versions of the language, variables are displayed by the term ‘fluents’ and are used not only in evaluating expressions, but also to represent quantities which may change over time as a result of some effectors.

Constants represent objects whose values remain unchanged and are used in the location of some arguments of action models stated in a domain. The relationships between objects in a domain are expressed through predicates. The predicates have an arbitrary number of arguments, and the order of these arguments and the type of each (if any) are clearly defined. The predicates are used as conditions or results of actions and to describe the state of the world at any given time.

Actions allow transition between two successive states. The transitions mean the addition or removal of predicates. The declaration of an action includes the parameters and variables used, and the conditions which should be valid in order to apply the action, i.e. the predicates that describe the state of the world..

PDDL offers two ways of constructing the results of an action. The results may either be a list of predicates or an expansion, but not both simultaneously. If the results of an operator are a list of predicates, which may be conditional or universally quantified, then they express the changes that would be introduced by the operator in the state of the world after the action is applied. More specifically, the results may include both predicates to be added and predicates to be removed from the state of the world. Predicates are not listed among the results of actions if they are considered to be unaffected by that action. If the results are expressed as an operand extension, then the

operand is decomposed into simpler parts, which can be in any combination of serial or parallel. This approach is used when the solution of a problem is expressed in the form of a series of operands instead of a set of predicates which represent the final state.

The axioms, in contrast, express relationships between successive statements that expressing relationships between facts in the same situation. The necessity of axioms arises from the fact that the definition of an action may not report all the predicates that are affected by that action. Therefore, additional predicates are deduced from the axioms after applying any operator. These are called derived predicates and are quite different to the elementary predicates. In newer versions of the PDDL language, the term ‘derived predicates’ has replaced the axioms term, but the concept and the idea behind them are the same.

The definition of a problem in PDDL includes the initial state and the final state (also to as the goal). The initial state includes a list of predicates, which are considered to be true and valid. It depends on the theory of the closed world, i.e. anything not mentioned as true in the original state is by definition false. The description of the final state can be exist either as free functions of first order predicate logic, or as an extension operand.

The PDDL 2.1 (Fox and Long 2003) was developed because of the need for a language capable of expressing the temporal and numerical properties of modern planning domains and problems. However, it was designed to be compatible with previous versions of the language and to maintain the basic principles.

The first extensions introduced numeric expressions. Elementary arithmetic expressions are functions that associate a number of domain objects with numerical values. Complex arithmetic expressions are formed from simple expressions by using

arithmetic operators between them. In order to support arithmetic expressions, additional information must be appended into the language. Such functions and hypothetical expressions include comparison between pairs of numerical expressions and values for operators, the assignment, increase and decrease. Assigning values or numeric values to functions are not part of the domain, so they should be declared in the problem definition. Plan metrics are another notable addition to the language. This allows the user to determine how each plan should be assessed and enable the ranking of solutions to identify the one which represents the best plan of action.

A further extension to this version of the language is the term ‘durative actions’, which are divided into discrete and continuous. While all previous versions of actions are considered instantaneous, this extension gives actions the ability to have a predefined duration. To work properly, the actions of the planning system are further extended to include associated timestamps (temporal annotations) to the conditions and the effects of actions. A condition associated with temporal annotation needs to be applied at the beginning, end or throughout the duration of the action. Similarly, an effect may be immediate or delayed, i.e. taking place at the beginning or at the end of the action duration.

The PDDL 2.2 (Edelkamp and Hoffmann 2004) added the derived predicates which were mentioned earlier, as well as initial delayed predicates (timed initial literals). These are events that are true or false at any given time as specified in the context of the problem. They are unaffected by the operators. The planning system indicates that these predicates should be applied when the action activates.

PDDL 3.0 (Gerevini and Long 2005) attempted to enrich the language structure by adding the ability to increase the expressive definition in the context of metrics that

determine the quality of plans. One of the ways of making this possible is the separation of hard and soft restrictions. Hard restrictions are required to be satisfied by the solution of the problem, while soft restrictions are desired (not required) to be satisfied by the solution of the problem. There are state trajectory constraints that place restrictions not only in the final state, but throughout the action sequence which is included in a plan. State trajectory constraints assert conditions that must be met by the entire sequence of states visited during the execution of a plan. They are expressed through temporal modal operators over first order formulae involving state predicates.

3.3 Knowledge Representation Methods

In the field of knowledge representation and reasoning, research usually focuses on developing methods for providing high-level descriptions of the world. Therefore, these methods can effectively be used to build intelligent applications to support, for example, the robotic system in accomplishing its tasks. When this system has the ability to discover implicit consequences of its explicitly represented knowledge, then it is characterised as knowledge based system.

In this study, the description logics (DLs), a family of logic-based knowledge representation languages, are chosen to represent the knowledge of an agent domain in an organised and formal manner. The most effective application of DLs is, however, becoming the basis of building the ontology of knowledge based systems in the Web Ontology Language (OWL) format.

The following subsection will describe the details of DL features and the benefits of using DLs to represent the system ontology.

3.3.1 Description Logics (DLs)

Description Logic languages are viewed as the core of knowledge representation systems as they consider both the structure of a DL knowledge base and its associated reasoning services.

Description logics (DLs), as explained by Baader et al. (2010), are knowledgeable fragments of first order logic which are intended for representing and managing knowledge. They are used to represent domain knowledge of applications through the specification of the domain concepts and the relationships between these concepts (concepts also known as terminologies). Assertions about the properties of, and relations between, individuals which are present in the domain are used to describe the agent world. The ability to reason and infer implicit knowledge from the explicitly represented knowledge is an important characteristic of description logics (Baader et al. 2008).

In DLs style, unary predicates represent concepts (i.e. sets of individuals that have common properties), while binary predicates represent relationships between individuals. Concept expressions can be built using a small set of connectives and constraints over the individuals that are in a relationship with a specific individual. Concepts that are not defined in terms of other concepts are called atomic concepts.

For example, to define the concept of "A room that contained a fridge, and all of its stuffs are either perishable or non-perishable." This concept can be described with the following concept description:

$Place \sqcap \neg Object \sqcap (\exists isContained.Fridge) \sqcap (\forall hasStuff.(Perishable \sqcup Non\text{-}Perishable))$.

This description employs several Boolean constructors: conjunction (⊓), which is interpreted as set intersection; disjunction (⊔), which is interpreted as set union; negation (¬), which is interpreted as set complement; the existential restriction constructor (∃*r*.*C*); and the value restriction constructor (∀*r*.*C*). An individual, for example *Kitchen1*, belongs to ∃*isContained*.*Fridge* if there exists an individual that is contained in *Kitchen1* (i.e., is related to *Kitchen1* via the *isContained* role) and is a *fridge* (i.e., belongs to the concept *Fridge*). Similarly, *Kitchen1* belongs to ∀*hasStuff*.(*Perishable* ⊔ *Non-Perishable*) if all its stuff (i.e., all individuals related to *Kitchen1* via the *hasStuff* role) are either perishable or non-perishable.

The present study has employed description logics to encode and reason about semantic domain knowledge for the purpose of plan generation (as will be described in Chapter 4). The major advantages of using description logics (DLs) are as follows:

- DLs provide a summarised representation of the world, as they can express general knowledge about classes of objects. Thus a lot of information can be kept implicit. For instance, one does not have to state explicitly that room *r5*, which is an office, contains a desk. Such information can be inferred from the general description of the class *Office*.
- DLs are quite expressive and supported by efficient inference mechanisms, making them practically useful.

Description Logics distinguish between terminological knowledge, the so-called TBox, and assertion knowledge, the ABox. The TBox contains definitions of concepts, for example the concepts *Action*, *Room* or *Oven*. These concepts are arranged in a hierarchy called taxonomy, using subclass definitions that describe, for example, that a *Kitchen* is

a specialisation of *Room*. The ABox contains individuals that are instantiations of these concepts, e.g. a particular *Table*, *table1*, as an instantiation of the concept *DiningTable*.

When modelling knowledge in robotics, the ABox usually describes detected object instances which are stored in the robot's knowledge base or evidence base. The TBox, in contrast, describes classes of objects or actions. Note that the differences between ABox and TBox are not specific to a domain or dependent on the environment. Therefore, individuals in the ABox can be very general, like an instance of *SpatialThing*, while classes in the TBox can be very specific, like the class of action "Grasping a milk box from the fridge with the right hand using a gripper".

In TBox, the relevant thought of a robot domain can be described by stating properties of concepts and roles, and relationships between them. In a TBox, a statement can be used to introduce an abbreviation for a complex description. For example, the name *HouseKitchen* can be used as an abbreviation for the concept description from above:

$HouseKitchen \equiv Place \sqcap \neg Object \sqcap (\exists isContained.Fridge) \sqcap (\forall hasStuff.(\ Perishable \sqcup Non\text{-}Perishable))$.

More expressive TBoxes allow the statement of more general axioms such as

$\exists hasStuff\ .Fridge \sqsubseteq KitchenObject.$

This says that only kitchen objects can have a fridge. This statement does not define a new concept, it just constrains the manner in which concepts and roles (*KitchenObject* and *hasStuff*) can be explained.

The ABox is used to describe a concrete situation by stating properties of individuals. For example, the assertions *HouseKitchen*(*Kitchen1*), *hasStuff*(*Kitchen1*,*Fridge1*), and

¬*Object*(*Kitchen1*) state that *Kitchen1* belongs to the concept *KitchenHouse*, and that *Fridge* is one of its objects. The underlying idea is that the ABox can then be used to represent knowledge that is not expressible in the restricted TBox formalism.

3.3.2 Ontology Layout

A knowledge representation method that depends on the taxonomy of concepts and relations between them is called "ontology". Ontology allows conclusions to be drawn using logical inference by the knowledge processing system. This is the essential objective of knowledge representation.

The Web Ontology Language (OWL) (Motik et al. 2009) is used for storing Description Logic formulas in an XML-based file format. OWL was originally developed for representing knowledge in the Semantic Web (Berners-Lee et al. 2001). Since that time, it has become a commonly used knowledge representation format. OWL is a file format for storing and exchanging description logic formulas. The naming in OWL differs slightly from that of DLs: "concepts" are usually called "classes" in OWL, "roles" are called "properties", and "individuals" are called "objects" or "instances". Table 3.1 lists the language constructs in DLs and their counterparts in OWL (Baader et al. 2008).

An OWL ontology can be seen to correspond to a DLs TBox together with a role hierarchy, describing the domain in terms of classes (corresponding to concepts) and properties (corresponding to roles). An ontology consists of a set of axioms which make it possible to assert subsumption or equivalence with respect to (i) classes or properties, (ii) the disjoint between classes, and (iii) the equivalence or non-equivalence of

individuals. Additionally, OWL allows features of properties (i.e., DL roles) to be asserted as transitive, functional, inverse functional or symmetric.

Table 3.2 compares the language axioms in DLs with their counterparts in OWL (Baader et al. 2008).

Table 3.1: A Comparison Between OWL and DL Constructors

Constructor	**DL syntax**	**Example**
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	¬Male
oneOf	$\{x_1 \ldots x_2\}$	{john, mary}
allValuesFrom	$\forall P.C$	∀hasChild.Doctor
someValuesFrom	$\exists r.C$	∃hasChild.Lawyer
hasValue	$\exists r.\{x\}$	∃citizenOf.{USA}
minCardinality	$(\geq nr)$	(≥ 2 hasChild)
maxCardinality	$(\leq nr)$	(≤ 1 hasChild)
inverseOf	r^-	hasChild^-

OWL has several advantages, which include the structured descriptions, the decision ability, and the fact that it is standardised and widely used. Despite these advantages, OWL suffers from some restrictions with respect to the representation of the knowledge of an autonomous robot. As a description logic language, it is limited to binary predicates, and stores all knowledge in terms of Subject-Predicate-Object triples. If

more complex n-ary relations are to be expressed, one has to create an intermediate instance (a parsed collection), that represents the relation to be expressed.

Table 3.2: A Comparison between OWL and DL Axioms

Axiom	**DL syntax**	**Example**
subClassOf	$C_1 \sqsubseteq C_n$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_n$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameAs	$\{x_1\} \equiv \{x_2\}$	{Pres_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
TransitiveProperty	P transitive role	hasAncestor is a transitive role
FunctionalProperty	T $\sqsubseteq (\leq 1\ P)$	T $\sqsubseteq (\leq 1$ hasMother)
InverseFunctionalProperty	T $\sqsubseteq (\leq 1\ P^-)$	T $\sqsubseteq (\leq 1$ isMotherOf$^-$)
SymmetricProperty	$P \equiv P^-$	isSiblingOf $\equiv$ isSiblingOf$^-$

Though this is less elegant than native support for such relations in the language, it generally presents no limitation. For example, the relation between *Place* and *Fridge* would be written as

```
<owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Place"/>
          <owl:Class rdf:about="#Fridge"/>
     </owl:intersectionOf>
</owl:Class>
```

while (>2 *hasStuff.Thing*) would be written as

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#hasStuff"/>
     <owl:minCardinality
          rdf:datatype="&xsd;NonNegativeInteger">2
     </owl:minCardinality>
</owl:Restriction>
```

In chapter 4, DLs and OWL ontology will be used to build the semantic knowledge base of the robot planning system. OWL format will then be used to represent the robot actions in an ontological manner.

3.4 Graphical Models

This section introduces basic notions of Probabilistic Graphical Models (PGMs). It describes two graphical models, namely Markov Networks (MNs) and Markov Logic Networks (MLNs). Their related formulas will be presented, along with examples of how MLNs serve as templates for constructing MNs. Chapter 5 will describe existing algorithms for learning and inference within MLNs.

Although classical logics provide a flexible framework for the representation of knowledge, they essentially ignore the fact that, under real-world conditions, knowledge is, to a large degree, subject to uncertainty. The class of graphical models is frequently used as a representational paradigm for representing uncertain knowledge.

The dependencies between variables, which usually reflect some knowledge and aspects of the real world properties, are conveniently represented by graphical models. Moreover, most aspects of the world can be sufficient to model the dependencies of its variables, so graphs of such dependencies are particularly desirable. It is easy for

humans to semantically visualise dependencies between nodes in graphs as edges, which facilitate understanding. Therefore, the modelling process of the knowledge in the problem is more readable. The knowledge structure should be expressed explicitly in intelligent systems in order to allow algorithms to exploit this structure to increase efficiency.

Probabilistic graphical models, such as Markov random fields and Bayesian networks (Koller and Friedman 2009) are widely used for representing statistical knowledge. In short, these models represent the probability distributions of system variables in a brief manner by taking advantage of independence. In the field of logic, graphical models are referred to as restriction networks and they are commonly used to represent the structure of constraint satisfaction problems.

3.4.1 Markov Networks (MNs)

A large joint probability distribution can be broken down into a set of smaller components within local views. These components can then be combined by considering relative dependency measures, which concern only a subset of system's random variables, in order to obtain a global probability measure. Locally, a (non-negative) numeric value is assigned to each particular configuration of each considered variable. This value has meaning only in relation to other such values and is proportional to the degree to which that configuration appears.

A graphical structure is implied between the considered variables within a local view. If two variables appear within a local view, then an edge will connect them. The framework of Markov networks formalises the way in which a full-joint probability

distribution can be specified based on local views and defines the relationship between conditional independence and the graphical structure that is implied by the local views under consideration.

Formally, a (discrete) Markov network or Markov random field (MRF) is a tuple $M = \langle X, D, G, \phi \rangle$ representing a joint probability distribution over a set of random variables $X = \{X_1, \ldots, X_N\}$ with corresponding (discrete) domains $D = \{D_1, \ldots, D_N\}$. The Markov network is an undirected graph $G = \langle X, E \rangle$ which contains one node for every random variable in X, while its set of edges $E \subseteq \left\{ \{X_i, X_j\} \middle| X_i, X_j \in X, i \neq j \right\}$ indicates dependencies between random variables. G is called the structure of the model. Since the graph is undirected, MRFs belong to the class of undirected graphical models (Koller and Friedman 2009).

The numerical sides of the distribution represented by M are given by a set of "potential functions" ϕ. The model has a potential function for each clique in the graph, where each potential function $\phi_k \in \phi$ maps from the fully connected subgraph (clique) in G to the non-negative real numbers, i.e. if $C_k = \{X_1, \ldots, X_{Nk}\}$ is the k_{th} clique in G, then

$$\phi_k : D_1 * \ldots * D_{N_k} \rightarrow \mathbb{R}_0^+ \tag{3.1}$$

The combination of the range of the functions ϕ_k constitutes the model's set of parameters. The set of possible assignments of random variables to values (the set of possible worlds), which is denoted by $\mathcal{X} := \mathrm{dom}(X) = \prod_{i=1}^{N} D_i$. M, represents a distribution over $\mathcal{X}$ as

$$\mathrm{P}(X = x) = \frac{1}{\mathrm{Z}} \prod_{k=1}^{|\phi|} \phi_{\mathrm{k}}(x_{\{k\}}) \tag{3.2}$$

Where $x_{\{k\}}$ corresponds to the state of the k_{th} clique in *G*. *Z* is a normalisation constant and is given by

$$Z = \sum_{x \in X} \prod_{k} \phi_{\mathrm{k}}(x_{\{k\}}) \tag{3.3}$$

The Markov blanket of a random variable X_i is defined precisely as the set MB(X_i) of random variables that must be given for X_i to be independent of all other random variables $X \setminus \mathrm{MB}(X_i) \setminus \{X_i\}$.

A new piece of information can be flexibly acquired by an intelligent agent when the probabilistic model represents a full-joint distribution over a set of random system variables. When the model probabilities of assigning particular random variables changes, then its beliefs should also change, leading to a belief update.

The process of obtaining new information from old information, which may be stored in the knowledge base or evidence base, is called inference. The most common two inference tasks are:

i. The computation of posterior marginal, i.e. the calculation of the conditional probability distribution over dom(*Q*) (the query) given a known assignment $E = e$ (the evidence) for two different subsets $\boldsymbol{Q} \subset \boldsymbol{X}$ and $\boldsymbol{E} \subset \boldsymbol{X}$. By introducing a new set (*U*) with $U := X \setminus E \setminus Q$, the posterior probability of an assignment $Q = q$ can be computed as

$$\mathrm{P}(Q = q \mid E = e) = \frac{P(Q = q, E = e)}{P(E = e)} = \frac{\sum_{u \in dom(U)} P(Q = q, E = e, U = u)}{\sum_{u \in dom(U)} \sum_{q' \in dom(Q)} P(Q = q', E = e, U = u)} \tag{3.4}$$

ii. The determination of the most probable assignment to a subset $\boldsymbol{Q} \subset \boldsymbol{X} \setminus \boldsymbol{E}$ given an assignment E = e, i.e. the computation of the *maximum a posteriori (MAP) hypothesis*. With set $U = X \setminus E \setminus Q$, it is given by

$$\underset{q \in dom(Q)}{\arg max} \sum_{u \in dom(U)} P(Q = q, U = u, E = e) \tag{3.5}$$

A related inference problem is the computation of the possible world, given the evidence $E = e$, which has the highest probability, i.e. the determination of the most probable explanation (MPE) of the evidence:

$$\underset{x \in \mathcal{X},\ x \vDash E = e}{\arg max} P(X = x) \tag{3.6}$$

As an alternative to potential functions, a Markov network can be represented as a log-linear model $\langle X, D, G, f, w \rangle$ where X, D, and G are defined as above and the potential functions ϕ_i are replaced by weighted features. Here a feature is a property that is either present or absent (either true or false) in a given world $x \in \mathcal{X}$. Thus f is a vector of feature functions $f_i: \mathcal{X} \to \{0,1\}$, and w is the associated vector of real weight values. The distribution over $\mathcal{X}$ is given by

$$P(X = x) = \frac{1}{Z} \exp(\sum_i w_i . f_i(x)) \quad (3.7)$$

3.4.2 Markov Logic Networks (MLNs)

Markov logic (Richardson and Domingos 2006) is a formalism that benefits from simplicity and hence received much attention and publicity in recent years. MLNs have high-level learning and reasoning capabilities, so they are a prime candidate for equipping with a technical system.

Markov Logic is a combination of MNs and first order language (FOL). A knowledge base (KB) is a set of (FOL formula) hard constraints on the set of possible worlds. In this case, any worlds that violate even one formula have zero probability of existing. Markov Logic is based on the idea that these constraints must be softened, i.e. when a world violates one formula in the KB it is less probable, but not impossible.

If a world violates fewer formulas, then it is deemed to be more probable. Each formula in Markov Logic has an associated weight that reflects how strong a constraint it is. The weighted formulas define a template for the construction of a graphical model that specifies a distribution over possible worlds. When the weight is high, this means a great difference exists between the log probability of a world that satisfies the formula and one that does not.

The ground Markov logic network specifies a probability distribution over the set of possible worlds *X* as follows:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|L|} w_i n_i(x) \right) \tag{3.8}$$

Where L is the Markov logic network language, w_i is the formula weight, $n_i(x)$ is the number of true grounding of formula in the world x, and Z is a partition function, defined as:

$$Z = \sum_{x' \in X} \exp\left(\sum_i w_i n_i(x') \right) \tag{3.9}$$

The details of MLN components, the main inference and learning algorithms will be presented in Chapter 5. The way that MLN can be used to solve the problem of planning under uncertainty is also discussed.

3.5 Planning System Architecture

This section will give an overview of the main components which constitute the proposed planning system in this study. Moreover, the interplay between these components is described in order to better understand their role in the system. Figure 3.3 groups the components by their functionality.

The central component is the knowledge representation that provides the mechanisms to store and retrieve all the different kinds of information in the system. Robots need very powerful knowledge representations that are expressive enough to describe all aspects of actions, objects, their properties and relations.

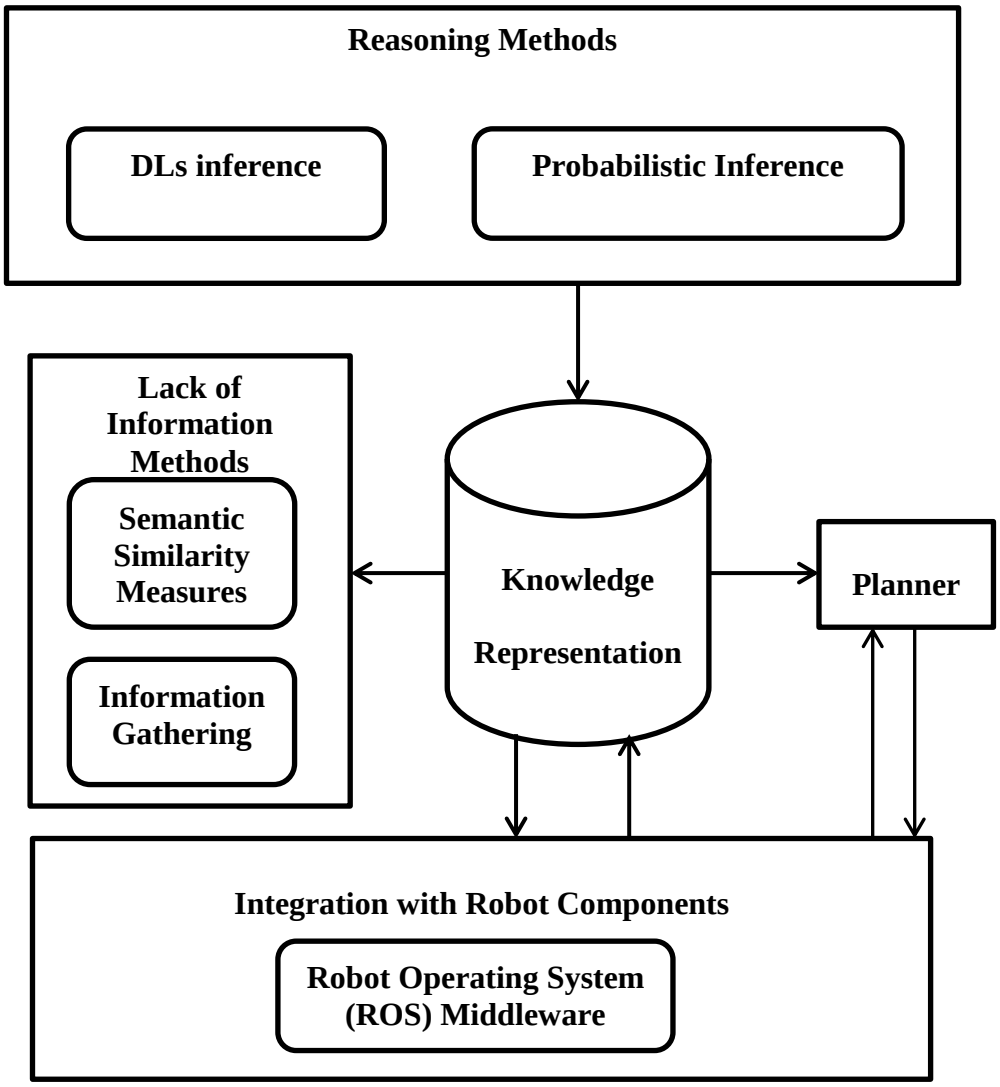


Figure 3.3: Planning System Architecture

The ‘reasoning methods’ component is used to enable the robot to derive new statements from the existing knowledge. Multiple general and special-purpose inference methods are used in this system. Description logic inference is the basic mechanism. It mainly deals with the types of things, and the automatic classification of things based on their properties. While pure description logic inference is completely deterministic, it is often desirable to represent uncertain information. The Markov Logic Networks allow the drawing of probabilistic inferences that combine the expressiveness of first-order logics with the representation of uncertainty.

Another module deals with situations of lack of information by using two techniques: similarity measures and information gathering techniques. Measuring semantic similarity between concepts enables a description of how close two concepts are in the knowledge base. This similarity value can for instance be used to determine appropriate storage locations for an object, namely where similar objects are stored. The information gathering techniques are used to enrich the robot world state with new information in order to recover the planning system from the incomplete information situations.

Integration with robot component is used to interface to other parts of the robot's control system. On the one hand, it updates the belief state inside the knowledge base based on external information; on the other hand, it offers knowledge and reasoning services to other components. This system can access information via the ROS communication middleware and provides this information into the reasoning process. This system also provides a query interface that allows other components to send queries via a ROS service.

3.6 Plan Generation under Deterministic and Probabilistic Conditions

The process of plan generation can be divided into deterministic and probabilistic types. This section will briefly describe each of them, while more detail will be added in Chapters 4, 5, 6 and 7 when presenting the contributions of this thesis.

In plan generation in deterministic conditions, the planner depends on the robot action model details to generate plans. The action details consist of parameters, preconditions

and effects parts. These details are deterministic, which means that the conditions, the necessary actions, and their consequences are clear and fixed. .

While in plan generation in probabilistic conditions, the planner also depends on the action model details to generate plans when uncertainty exists in the world states (action conditions), and in the effect of the actions. The probabilistic planning also depending on the MLN unit in order to handle situations suffered from uncertainty in action outcomes and robot world states.

Some of the related tools which are used in the present study are already available, but other important tools are created to cover problems which are not addressed by the available tools. Chapters 4, 5, 6 and 7 will present these tools as direct contributions of this work.

3.7 Robot and Environment Specifications

The aim of this section is to describe the proposed Multi-Role Shadow Robotic System for Independent Living (SRS)[1] its environmental specifications that will be used in the test scenarios.

i. Multi-Role Shadow Robotic System for Independent Living (SRS) is selected as a proposed robotic system for testing scenarios. SRS robot is remotely-controlled, semi-autonomous robotic system which is intended to work in a domestic environment to help elderly people. The Care-O-bot®3 (Reiser et al. 2009) robot is

[1] http://srs-project.eu/

selected for the SRS project. The Care-O-bot®3 robot has the following specifications:

1- Actuators and mobility: The system is able to handle general sized household objects and door handles. Care-O-bot® 3 is equipped with a highly flexible, commercial arm with seven degrees of freedom as well as with a three-finger hand. This makes it capable of grasping and operating a large number of different everyday objects. Using tactile sensors in the fingers, Care-O-bot® 3 is able to adjust the rasping force.

2- Basic navigation and manipulation planning: The manipulators of Care-O-bot®3 can be actuated by a simple control interface which allows movements in joint and Cartesian space. According to the parameters given in the configuration files, the robot is modelled by oriented bounding boxes, which are used for collision avoidance calculations.

3- Basic environment perception: A multiplicity of sensors enables Care-O-bot® 3 to detect the environment in which it is operating. These range from stereo colour cameras and laser scanners to a 3D range imaging camera. The sensors serve, for example, to detect and locate objects for manipulation as well as relevant obstacles in the robot's environment.

ii. SRS Environmental Specifications: Environmental conditions are a key factor in independent living for elderly people who prefer to live in their homes as long as they can. The environment in which SRS will operate will be the elderly person‘s home. As it is normal in any home, the master of the house will not be the only human present in it.

Occasionally relatives or neighbours may visit the elderly, possibly with children. It may be that the elderly person is occasionally caring for his grandchildren, so they are present in the house. Other people also may occasionally be present, such as people coming to the elderly person for offering services (nurse, delivery boy, carrier, postman, etc.). The main objects in the elderly people environment are: washing machine, dish washer, TV, telephone, etc.

3.8 Summary

This section has reviewed the theoretical basis and the planning domain and problem formalism then the methods used to represent the semantic knowledge and the layout of this knowledge. The review has highlighted the following points:

1. The planning domain and problem has been formalised by using planning domain and definition language (PDDL). This formalism has been the standard in the international planning competitions (IPC).
2. The PDDL has two structures: the first structure has been used to model the planning domain while the second structure has been used to model the planning problem.
3. The PDDL has used the STRIPS model to represent the main components of each structure that PDDL has.
4. The Description Logics (DL) has been used to represent the semantic knowledge, which consists of fragments of first order logic.
5. The Web Ontology Language (OWL) has been used to store the DL formulas in an XML-based file format.

6. The SRS robot is not embedded with planner to generate plans to accomplish its tasks. The novelty of this work is to engage a planner in SRS robot operation, whereby endowing the robot the ability to work autonomously.
7. The generation of planning domain and problem files in PDDL style is prepared manually in general and there is a need to generate PDDL files from a source of knowledge automatically. So, the work in this thesis (as it will be seen in chapter 4) deals with developing an algorithm to generate planning domain in PDDL style from robot knowledge base.

Having established the mechanisms of using the PDDL style, DL for representing the knowledge and OWL for storing the DL formulas, next chapter will use them to model the proposed planning system under deterministic conditions.

Chapter 4

Semantic Based Planning under Deterministic Conditions

4.1 Introduction

Knowledge is one of the most important tools enabling robots to provide a service in a domestic environment. This knowledge supports them in intelligently completing their service tasks in a real environment. A Semantic network (Berners-Lee et al. 2001), ConceptNet (Liu and Singh 2004) can be used to represent this knowledge. These techniques are capable of representing knowledge as concepts and the semantic relations between these concepts. The description of the concepts and the relationships between them is represented as ontology. The semantic information can improve robot reasoning and knowledge inference. Actions are applied based on the knowledge in order to change robot states, and these actions are also important.

First, the robot knowledge is represented as ontology representing the main concepts and the relationships between them. Then, the robot actions are formed as models which are also represented as ontology. This representation reflects action inputs, preconditions, outputs and effects. Their arguments are then connected to the robot knowledge base to enrich the planning domain with semantic facts about the objects and places in the robot's environment. Action models, in this sense, are somewhat similar to web services (Martin et al. 2004).

Having reviewed, in Chapter 2, research works regarding the generation of robot plans and strategies for dealing with unexpected situations, solutions to the problems addressed in this thesis will now be presented. In this chapter, a novel approach is

presented for the intelligent generation of symbolic plans by mobile robots acting in indoor environments such as offices and houses. The novelty of the approach lies in the use of domain knowledge to derive implicit information related to the robot world states and actions details. The robot planning system uses the immediately available evidence database to check whether those expectations are met or violated during the robot work.

Depending solely on explicit knowledge to construct an action sequence implies that the derived expectations are directly observable. For example, a mobile robot that has generated the planned action *move(robot,hall,r1)*, to enter the living room *r1*, would query its self-localisation system to provide it with the explicit expectation in order to hold *in(robot ,r1)* in its evidence base. In this manner, the plan generation process relies entirely on the accuracy of the self-localisation system. Moreover, checking expectations in real-world environments is an inherently complex process that goes beyond checking what the robot directly observed.

This chapter proposes to increase the reliability of plan generation by incorporating more advanced forms of reasoning. In particular, the semantic knowledge domain is proposed to derive implicit knowledge about robot world states and the effects of actions. Implicit knowledge means knowledge which can be logically derived from explicit knowledge (the knowledge encoded in the action model or world states) through the use of reasoning techniques. In the above example, if the action *move(robot,hall,r1)* succeeded, and since *r1* is an instance of the class *LivingRoom*, the robot should expect to validate information about objects that are typical of a living room (e.g. TV, sofa, etc.). If the robot sees an oven, it should conclude that it is not in the living room, so that *in(robot, r1)* cannot be held as true in the evidence database.

Another example is the action *grasp(robot,c1)* which instructs the robot to grasp a cup of coffee *c1*. Semantic knowledge could be queried to obtain the implicit knowledge that the object in the gripper has properties such as being a container and having one handle. Therefore, checking implicit expectations when acting in indoor environments can help, among other things, to verify that the robot is in the correct room, and (1) is not dislocated or (2) does not have an erroneous map.

Implicit expectation details would add complexity to the task planning if the task planner is required to reason about them. Therefore, they are extracted from the semantic knowledge base and action models, and encoded in a separate component. They are used only when necessary, i.e. outside the planning engine of the task planner.

The rest of this chapter is organised as follows: Section 4.2 presents a motivating example; Section 4.3 explains the robot semantic knowledge domain; Section 4.4 discusses the main components of the planning system framework; Section 4.5 gives an overview of the planning approach and its algorithm; and Section 4.6 presents the experiments which were used to validate the approach which is presented in this chapter. Following this, the main outcomes of this chapter are discussed in Section 4.7.

4.2 A Motivating Scenario

To better illustrate this chapter hypotheses, a scenario will be described where a mobile robot is operating in a house environment to accomplish a multitude of household tasks, such as (i) cleaning the floor, (ii) serving drinks to guests, and (iii) doing laundry, among others. Figure 4.1 shows a map of such a house where the robot can live and serve. The house comprises rooms of different types such as bedrooms, kitchen, etc., as

well as objects of different types that can exist in different rooms. The object types include sofas, beds, tables, chairs, kitchen appliances (oven, fridge, ...), plants, etc. The environment is not specifically designed to be structured in a way that makes it easy for the robot to act. The environment is also dynamic, as objects such as chairs and cups can be displaced from one location to another, either by the robot itself or by the humans living in the house without notifying the robot.

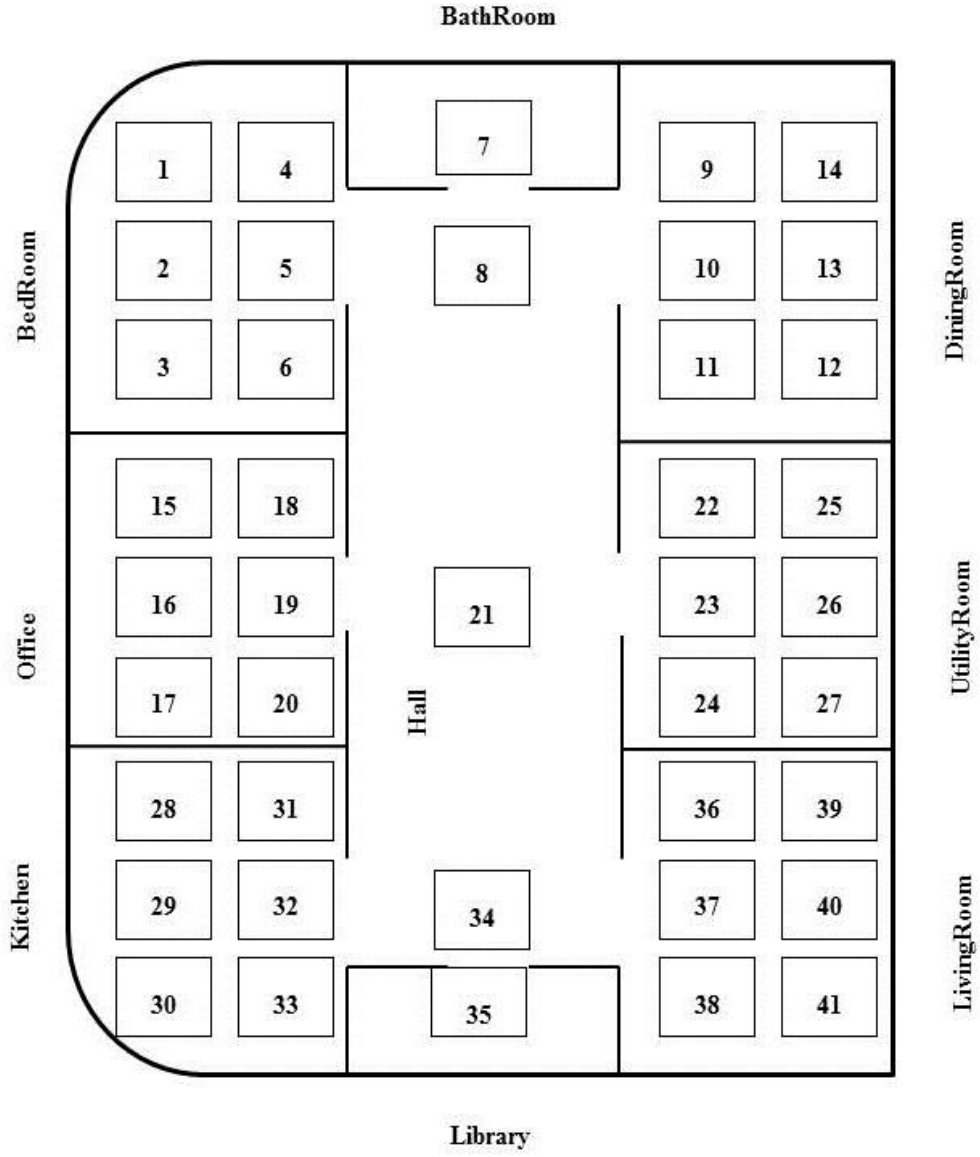


Figure 4.1: Robot Environment

The robot is supposed to act autonomously and is equipped with functionalities that help it to accomplish its tasks. These include low-level navigation and manipulation functionalities as well as high-level deliberation and problem solving capabilities. In particular, an on-board planning engine is used to synthesise plans that specify the actions required to accomplish a specific task. The robot is also granted access to a knowledge base (KB) where information about its environment is stored. Suppose that while the robot is busy cleaning the living room, it is asked to bring a cup of coffee immediately. To do so, the robot must first suspend the task of cleaning and then call its on-board planner to generate a plan that helps it to accomplish the assigned task of bringing coffee.

The task planning system uses information about the current location of the robot as well as the domain knowledge (e.g., cups are generally found in the cupboard, which is located in the kitchen) to generate a task plan. This plan could include the following actions:

move(robot,r4,hall),move(robot,hall,r5),open-cupboard(robot,cb1),take(robot,c1,r5),
fill(robot,c1,coffee),move(robot,r5,hall),move(robot,hall,r4),deliver(robot,c1,person1)'

where *r5* and *r4* are symbols denoting the kitchen and the living room respectively, while the cupboard is referred to by the symbol *cb1*, and the cup by the symbol *c1*. The final action specifies that the robot should deliver the cup of coffee to the person *person1* who ordered the cup of coffee. If the robot started from a position where it was initially dislocated, then the generation of action *move(robot,r4, hall)* could result in the robot being not in *r4* but in *r1* instead. A standard plan generator uses the available information to generate the action *move(robot,hall, r5)*, expecting that the robot will be in room *r5* based on the current location provided by the robot's self-localisation

system. Due to an initial error in orientation, the location of the robot can be erroneously computed to be *r5*.

Using semantic domain knowledge when generating plans makes it possible to provide the planning system with the implicit effects arising from being in room *r5*. In particular, since *r5* is asserted to be a kitchen, the robot expects to find indications that this room is a kitchen, such as seeing an oven, sink, or stove. If, for example, the robot sees a bed, it should conclude that it is not in a kitchen, but in a bedroom. Such information is derived from the semantics of the different rooms and the objects present in the house. In this case the robot can depend on the generated plan to know that the execution of the action *take(robot,c1,r5)* will be successfully executed by verifying not only that its gripper is holding something but also that the description of the object satisfies the object type *Cup*.

4.3 Robot Semantic Knowledge Base (Robot Environment Ontology)

As mentioned in Chapter 1, semantic knowledge refers to the meaning of objects expressed in terms of their properties and relations to other objects. Objects that share the same properties and relations are grouped into classes (or concepts). For instance, objects of type *Room*, and which include a bed, are instances of the class *BedRoom*. Similarly rooms with a sofas or a tv set are defined as the *LivingRoom* class. This reasoning captures the way that humans organise knowledge about objects as instances of general categories. Therefore, semantic knowledge can be used to help mobile robots communicate with humans. For instance, in the work of (Theobalt et al. 2002), instead of using metric data, a robot can ask humans about its location in terms of high-level

descriptions of locations. Semantic knowledge has also been used in other areas of mobile robotics, such as scene analysis (Hois et al. 2006) and map building (Galindo et al. 2005; Nüchter et al. 2006; Ekvall et al. 2007).

The semantic knowledge base should capture knowledge about the objects that form the robot's environment. The specification of such knowledge might initially appear to be a simple task. However, indoor environments are usually congested with objects of different types, which makes it difficult to provide knowledge about each of them.

In this chapter, the semantic knowledge is used for the purpose of generating symbolic semantic plans, and provides knowledge about the task at hand, i.e. objects in the generated plans. Therefore, the design of the semantic knowledge base takes into accounts the formal definitions and details of the actions (action components) that form the planning domains.

As a first step, the semantic action models are used to identify, in the parameter part, the objects type, which are manipulated by an action, along with their properties. Then, the knowledge about the different types of objects is provided to the planning system in terms of object properties and relations to other objects. Next, the same process is applied to add knowledge about the related objects, and so on. For instance, in a navigation planning domain, the action *move(robot,loc1,loc2)* is used to model the movement of the robot from location *loc1* to location *loc2*. Consequently, the semantic knowledge base includes knowledge about the different types of locations that exist in the robot's environment. These can be corridors, halls, and rooms. The knowledge base also includes information about the different types of room (e.g., kitchen, office, etc.), and also the typical objects of such rooms, such as ovens, desks, etc.

The knowledge base in this section is built to represent information that is clear for humans, but not explicitly clear for robots. This knowledge base will be used to support the task planner to build a more efficient and reasonable plan. The planner will take into consideration the semantics of objects and places that make up the robot world. The creation of a knowledge base is dependent on some known knowledge bases. The first of these is the Common-Sense knowledge base which has been collected and made publicly available by the Open Mind Indoor Common Sense project (OMICS) (Gupta and Kochenderfer 2004). It is intended to be used in indoor mobile robotics. The second known knowledge base is OpenCyc ontology (Lenat 1995), which was created by experienced experts with the intention of building an ontology to be a general standard for robot knowledge bases. It contains concepts forming an ontology in the domain of human consensus reality and assertions (facts and rules) using relations that interrelate, constrain and, in effect, (partially) define the concepts. The third known knowledge base is KNOWROB (Tenorth et al. 2010b) which has been used to support the KNOWROB knowledge processing system. All these knowledge bases are used in creating SKB in order to provide the robots with the broad range of knowledge necessary to accomplish their tasks.

The process of building the SKB is as follows:

1- From OMICS: the information that is clear for the human, but is not clear for the robot has been extracted and inserted in the SKB. For example, turning off the cooker after using it is clear for the human, but it is not clear for the robot and should be defined explicitly as a property of using the cooker. OMICS focuses on the kind of knowledge required by robots acting in indoor environments.

2- From OpenCyc: the SKB layout of the upper levels, including classes, their hierarchy and properties, has been adopted from the OpenCyc ontology. Adopting the ontological structure also means to adopt a certain way of thinking since the vocabulary means that a language provides shape the way how things can be described.

3- From KNOWROB: the modelling of objects and places in SKB is similar to the representations in KNOWROB, whereas other parts like the description of object locations and the place details have been developed specifically for SKB.

Figure 4.2 shows an excerpt of the knowledge that is used in this study and is represented as hierarchical levels which reflects the robot environment ontology. This knowledge base is stored as an OWL file to reflect the ontology of the environment that consists of rooms (such as the kitchen and bathroom), and objects (such as the fridge, TV, etc.). The knowledge about objects is restricted to be in terms of properties that the robot can directly observe or are defined in terms of other observable properties. For example, the definition of an object of type *Cup* might include that the object must be a

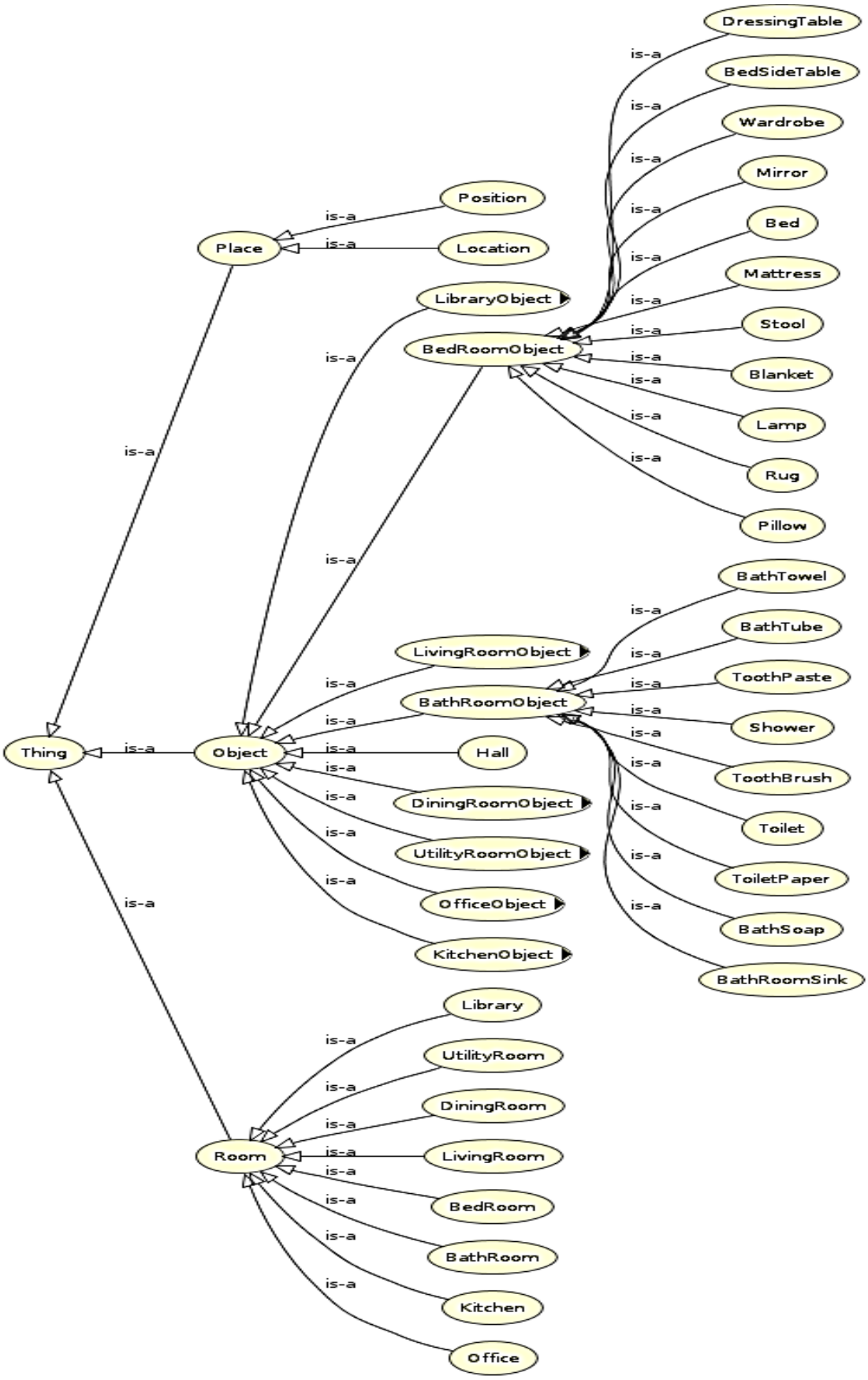


Figure 4.2: Robot Knowledge Ontology

container that has one handle. The container and handle are atomic concepts, and hence they must be directly observable. Direct observability implies that it is the task of the perception module to tell whether a perceived object is an instance of an atomic class, e.g. whether a perceived object is a container. All these details are stored in the robot evidence database.

The most important sections in the ontology are the *Room*, which contains descriptions of the main rooms in the robot environment, and *Object*, which describes abstract concepts such as *LivingRoomObject* or *KitchenObject*, as well as all the different object classes. Most objects in the robot's environment, together with pieces of furniture or body parts, are subsumed under the Object class. Another notable branch is the Place class, which describes locations and positions in the robot map.

The following example (which is based on Figure 4.2 and Protégé[2] style) describes what is meant by atomic classes and defined classes. The roles are used to restrict the extension of certain classes.

Atomic classes: *Room, Object*.

Defined classes: *Kitchen isA Room* and *isContained KitchenObject*

KitchenObject isA Object and *isLocatedAt Kitchen*

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

[2] http://protege.stanford.edu

After the general semantic knowledge base is created, specific instances of classes can be retrieved by creating a query which including the instances properties that reflect the state of these instances in the environment. For example:

SELECT ?k1 ?f1

WHERE {?k1 rdf:type skpg:Kitchen

?f1 rdf:type skpg:KitchenObject

?k1 skpg:isContained ?f1}

returned that *k1* is an instance of kitchen and *f1* as an instance of *KitchenObject*. As a result of the property *isContained* 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 *Kitchen*. 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 added to the knowledge base, they are classified according to what has been asserted.

4.4 Planning System Architecture

Figure 4.3 illustrates the main components of the proposed robot planning system. It consists of:

- The semantic action model (SAM), which contains the main robot actions and expresses the robot capabilities.
- The semantic knowledge base (SKB), which is described in Section 4.3.

- The ontology manager and the transformation algorithm, which convert SAM and SKB from OWL style into PDDL style.
- Problem and domain definition in PDDL style, which result from step3.
- The planner, which is responsible for generating a robot plan for a given task.

The next subsections present details for each of these components (except SKB which was described in the previous section).

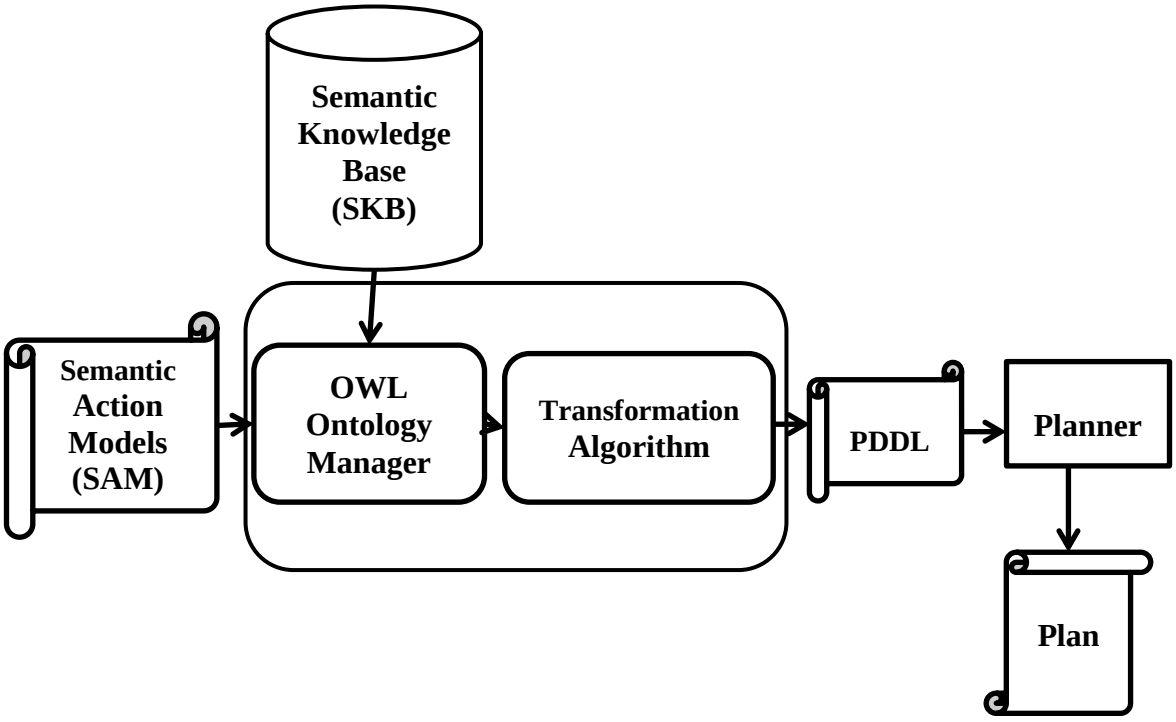


Figure 4.3: Planning System Architecture

4.4.1 Semantic Action Models (SAMs)

The domain definition begins with robot actions, which express the robot abilities. The action model is formed in STRIPS-style to represent its preconditions and effects. This model is stored as an OWL file that represents it as an ontology in order to connect

action parameters to the main ontology (knowledge base). The idea of SAM is clear in Figure 4.4 which explains the SAM model of the *move* action.

```
<process:Input rdf:ID="_ROBOT">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                        file:/project/ontology/world.owl#Robot</process:parameterType>
<process:at rdf:resource="#_FROM"/>
<rdfs:label></rdfs:label>
</process:Input>

<process:Input  rdf:ID="_FROM">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                                file:/project/ontology/world.owl#Place</process:parameterType>
<process:isConnected rdf:resource="#_TO"/>
<rdfs:label></rdfs:label>
</process:Input>

<process:Input  rdf:ID="_TO">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                        file:/project/ontology/world.owl#Place</process:parameterType>
<process:isConnected rdf:resource="#_FROM"/>
<rdfs:label></rdfs:label>
</process:Input>

<process:Output  rdf:ID="_ROBOT">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                        file:/project/ontology/world.owl#Robot</process:parameterType>
<process:at rdf:resource="#_TO"/>
<process:been-at rdf:resource="#_TO"/>
<process:not-at rdf:resource="#_FROM"/>
<rdfs:label></rdfs:label>
</process:Output>
```

Figure 4.4: Semantic Action Model for Move Action (SAM)

The main components of the proposed semantic action model, which are related to the process of domain generation, are ID, Inputs, Preconditions, outputs and Effects. These components are depicted in the Figure 4.5 (left side). These components are connected to the knowledge base in a semantic way by referring to the SKB which contains the class to which that component is related. For example, for the *move* action, the precondition of the this action is at(*robot*,*from*) and represented as input to the process "Input". The parameters *robot* and *from* are connected to the SKB by using the name of

the knowledge base and the name of the parameter's class. Then the SAM is stored as an OWL file (Bechhofer et al. 2004). The components of SAM are described as follows:

- Profile part shows an action as a function of three basic types of information: (i) what robot part provides this action, (ii) what function the action does, and (iii) what robot part hosts the features that specify characteristics of the action. The profile describes the action inputs and outputs, the preconditions required for the action to occur, and the expected effects that result from the execution of the action.
- Process part is a specification of the ways a robot may interact with an action. There are two types of processes: (i) the atomic process, which has one (or more) simple input(s) and one simple output, and (ii) the composite process, which takes more than one input and produces one or more outputs. The process part is the main part of the action model. The algorithm that operates on it will be presented in the next section. The process inputs and outputs and the world states are used to produce new information, while preconditions and effects can produce change in the robot world as a transition from one state to another state.

The process inputs represent information that is required for operating the process. The process outputs represent the information provided to the user or the next action(s). The process preconditions should all be verified in order for the process to be invoked successfully. The process effects show the results of applying the actions.

There are some typical properties which are used to describe process functions and their parameters. These properties are: hasInput (used to describe an action's inputs); and hasOutput (used to describe an action's outputs). Sometimes, when necessary, other

properties are used to describe the action preconditions hasPrecondition and effects hasEffect.

Figure 4.5 represents the matching process between the SAM in OWL format and SAM in PDDL format. The ID in OWL will be the name of the action in PDDL, the inputs and Preconditions in OWL will be the preconditions of the action in PDDL then the Outputs and Effects in OWL will be the effect part of the action in PDDL.

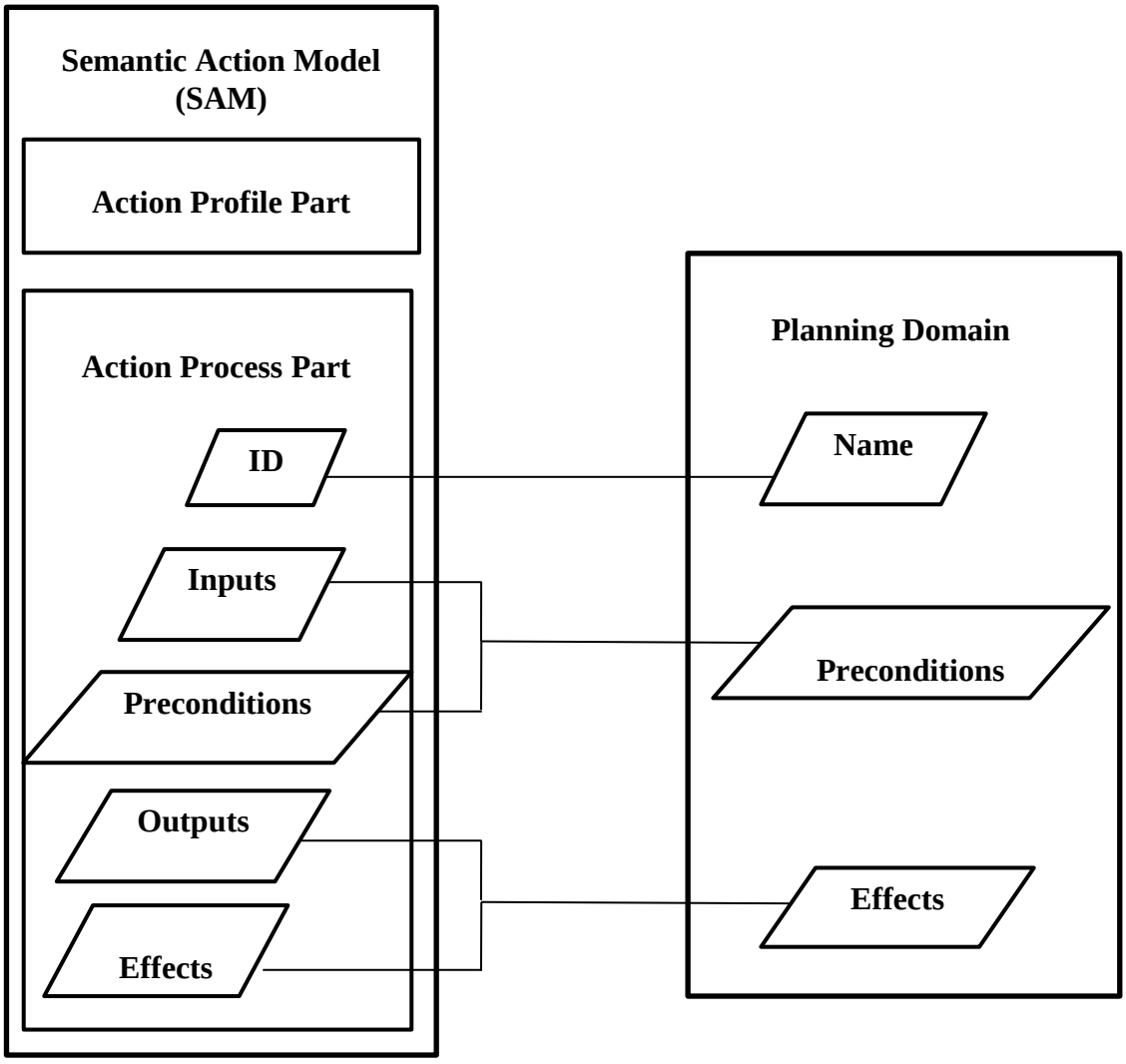


Figure 4.5: Mapping between Semantic Action Model and Planning Domain

4.4.2 Planner

The planner is at the core of the planning system. It takes as it's input(s) the problem and domain definitions in PDDL format. This format became a standard for planning competitions since 1998. There are many different planners which have been developed. In this thesis, two types of planners are used to produce the plan. The first is known as Metric-FF (Hoffmann and Nebel 2001) and the second is LPG-td (Gerevini et al. 2003). These planners have become very popular during the last few years and have received awards for best planner at several international planning competitions (IPCs). The work in this study is not limited to these planners; any STRIPS-style planner can be used. The Metric-FF and LPG-td planners are already undergoing redevelopment to improve their computational efficiency in planning, so the overall improvement of the proposed framework can be high.

Metric-FF is an open source implementation of the c++ actions within the design algorithm, Graphplan (Blum and Furst 1997), which follows the STRIPS formalism. It uses the advantages of efficient graph algorithms to reduce the search space and provide better solutions. The LPG-td is the new, expanded and improved version of LPG. It is based on local search and graphic design effects, and can produce good quality plans. An important advantage of LPG-td is that it can produce more than one plan. Therefore, for problems with high complexity, a greater time left to run means a greater chance of finding an optimal solution (assuming such a plan can exist).

4.4.3 Semantic Action Model Transformation to Planning Domain Definition Algorithm

This section deals with the algorithm that is used to transform SAM from OWL style to PDDL style. Figure 4.5 shows the mapping process between SAM and the planning domain. The algorithm uses that mapping process to generate a planning domain from semantic action models.

Algorithm 4.1: Pseudo code shows that the first step in the translation process is to read SAM in the OWL format. Then a loop is generated to read every action model in the list *M*. For each $m \in M$, the transformation process extracts the following information:

- A set of *m* parameters, which are stored in the variable *v*.
- A set of *m* preconditions, which are a conjunction of all inputs and precondition statements in *m*, which are stored in the variable *preco*.
- A set of m effects. These are a conjunction of all the output and effect statements in *m*, and are stored in the variable *effec*.
- A combination of the variables *v*, *preco*, and *effec* in PDDL format, stored in variable *P*. In the end this combination represents the planning domain.

Algorithm 4.1

Procedure SAM to PDDL

Input: M = SAM set

Output: PDDL planning domain set P

$P=0$

For each $m \in M$

v = input parameter group of m

preco = (and(conjunction of all inputs and preconditions defined in model *m))*

effect = (and (conjunction of all outputs and effects defined in model *m))*

Add *(m(v) preco effec)* to P

End

4.4.4 Problem Definitions

A planning problem is modelled according to the Stanford Research Institute Planning System (STRIPS) style (Fikes and Nilsson 1971) as a tuple <*I, A, G*> where *I* is the initial state, *A* is a set of available actions and *G* is a set of goals. States in STRIPS are represented as sets of atomic facts and predicates. Set *A* contains all the actions that a robot has the ability to do, and can be used to modify states. Each action A_i has three lists of facts containing (i) the preconditions of A_i, (ii) the facts that will be added to the state and (iii) the facts that will be deleted from the state. These are denoted as precondition(A_i), add(A_i) and del(A_i) respectively. The following points hold true for the states in STRIPS notation:

1. An action A_i is applicable on a state *S* if precondition(A_i) $\subseteq S$.
2. If A_i is applied to *S*, the generated state $\boldsymbol{S}^{-}$ is calculated as

$\boldsymbol{S^-} = (\boldsymbol{S} - \boldsymbol{del}(\boldsymbol{A_i})) \cup \boldsymbol{add}(\boldsymbol{A_i})$.

3. The solution to a planning problem (plan) is a sequence of actions, which, if applied to *I*, lead to a state $\boldsymbol{S^-}$ such that $\boldsymbol{S^-} \supseteq G$.

According to Figure 4.5, the following rules are used to collect the predicates for each of the lists above.

1. name(A_i) = SAM_i.id
2. $\text{precondition}(A_i) =$

   $\cup_{f=1}^{n} \text{SAM}_\text{i}.hasInput.Parameter_f \cup \cup_{f=1}^{m} \text{SAM}_\text{i}.hasPrecondition.Parameter_f$
3. $\text{add}(A_i) =$

   $\cup_{f=1}^{n} \text{SAM}_\text{i}.hasOutput.Parameter_f \cup \cup_{f=1}^{m} \text{SAM}_\text{i}.hasEffect.Parameter_f$

4.5 Overview of the Approach

This section will give an overview of how semantic domain knowledge can be integrated with semantic action models (SAMs). This generates the planning domain and problem definitions in the process of constructing symbolic plans for the given autonomous mobile robot tasks. Then, semantic knowledge base is used to extract the implicit expectations of action effects. The process is meant to be used for deterministic domains where both knowledge and action details are assumed to be expressed explicitly. This section starts by describing the overall process of generating plans and obtaining the implicit expectations of the effects produced by the actions. Then, an overview is given of the algorithm involved in deriving and checking the implicit expectations of the object in the action model.

4.5.1 The Overall Planning Process

The robot planning process tries to imitate the human decision making process. In planning, a solution to a problem is given in the form of a sequence of basic actions that transform a given initial situation of the environment into a desired or goal state. The purpose of planning is to synthesise an abstract trajectory in a search space (also named state space, since it consists of possible states of the world), predicting outcomes while choosing and organising actions of different types to reach goals.

As explained in Chapter 2, the generation of symbolic plans is a model-based process that exploits the explicitly modelled effects of an action to construct a sequence of actions that together perform a given robot task. The explicit outcome of an action is extracted from SAM using the OWL Ontology Manager module in the planning system. A model of the action can also be used, together with knowledge of the objects manipulated by the action, to compute a set of expectations that are not explicitly encoded in that model. These expectations can then be verified using the perceptual (evidential) information used to check the explicit effects.

There are two possible methods for using the information in the knowledge base with SAM to produce the necessary information in robot states and support the planner to generate the plans. The first method is to use the actions separately and then combine their effects to produce a final result (or decision) about whether the generation of the plan has succeeded. The second method is to use the result produced by the process of constructing the plans, based on the implicit expectations. This can be used as an additional check of the results produced by the process of generating plans from the explicit effects of the action.

Using the first method implies having a mechanism that can handle situations where the results of the two processes might be contradictory. For example, the process of obtaining the explicit effects of generating the action *move(robot,r4,r5)* could deduce that the planning has failed if, according to the self-localisation module, the robot is still in room *r4*. On the other hand, the process of recording the implicit expectations can declare that the room where the robot is located is a kitchen because an oven is observed this room.

In this chapter, the second method is used. The process of validating the generated plans is based on the implicit expectations, which are extracted from the action effects. This process is called in order to validate the explicit effects of the action in the world state that is produced by the current action. Hence, the final result of generating the plan actions is the result returned by the process of embedding the checking of implicit expectations of each action in the plan. Therefore, the schema of generating the plan action (A) in a world state (w_s) is comprised of the following steps:

1. The first step is a prediction, where the model of action (A) is used to compute the explicit effects of (A) when applied to (w_s).
2. In the second step, all the computed explicit effects are checked in the proposed resulting world state. If all the explicit effects are verified, then the implicit effects of generating (A) in (w_s) are derived and checked in the current world state as well. If one of the implicit expectations is found to be violated, a failure is returned to the plan generator. Otherwise, the generation of the action has succeeded, and success is returned to the plan generator.
3. If in the second step, some of the explicit or implicit effects were found to be violated in the current world state, then a failure is returned to the plan generator.

4. The plan generator carries on the generation of the rest of the plan only if it gets "success" from the step2. Otherwise a recovery procedure might be invoked to recover from the encountered unexpected situation.

In order to formalise the planning approach in this thesis, some basic concepts related to traditional AI planning are defined. These include logical predicates and planning states, which are the basis of classical logical planners. Next, an operation is defined for integrating planning states with action models.

4.5.2 Planning Basics

1- **Logical Predicate**: A logical predicate *pred* is an atomic sentence of a first-order finite language L with n parameters such that: $pred = (\gamma, Param)$, where γ is the predicate symbol that represents a k-ary function, and *Param* is an ordered list of k language constants which are the parameters of γ. Any components of a given predicate *pred*, γ (*pred*) and *Param*(*pred*) can be referred to.

2- **Planning State:** A planning state, or a state for short, is a finite and consistent set of logical predicates joined through two logical connectives: and ($\wedge$); or ($\vee$). This set represents some world information.

The set of all possible planning states over language L is denoted as Γ. Informally, planning is the process that transforms an initial state that represents the current information available from the world, into a goal state that represents the desired final situation. Given a planning state $S \in \Gamma$, a function that yields a set containing the distinct parameters from all logical predicates of S is defined as the State Parameters

function, *SP*(*S*). Similarly, a function that yields a set containing all distinct predicate symbols of *S* is defined as the State Predicate Symbols function, *SN*(*S*),

More formally:

$$SP(S) = \bigcup_{pred \in S} param(pred) \tag{4.1}$$

$$SN(S) = \bigcup_{pred \in S} \gamma(pred) \tag{4.2}$$

3- **Operator** (*O*): An operator is a pair < *Preco*, *Effec* >. *Preco* is a set of logical predicates representing the conditions under which the operator is applicable. *Effec* = (add, del) contains two sets of logical predicates add and del that will be added and removed respectively from the planning state in which the operator is applied, in order to obtain a resulting state.

Thus, given a planning state *S*, an operator *O* can be applied if and only if the set of logical predicates of its preconditions are true under *S* through a given instantiation of the parameters of *Preco*(*O*). After the application of the operator, state *S* is transformed into S^-, using the set of effects *Effec*(*O*) instantiated in the same manner:

$$S^- = [S \cup \text{add}(O)] - \text{del}(O) \tag{4.3}$$

4- **Problem Space, Planning and Plan:** A problem space is composed of a first-order language (*L*), an initial state (S_i), a goal state (S_g), and a set of finite operators (*O*). Within a certain problem space, the planning process consists of searching a chained

sequence of operators, o_1, o_2, . . . , o_n, that transforms the initial state S_i into the goal state S_g. Such a sequence of operators is commonly called a plan.

4.5.3 Semantic Plan Generation Process

A process for plan generation based on semantic knowledge is outlined in Algorithm 4.2. The process typically checks if a planning time object fits the expected description of an object. For instance, if the action to be inserted into the plan is *grasp (robot,b1)* to grasp object *b1* (the planning time object), then the object that will actually be manipulated by the action grasp is the expected object; indeed, this needs to be checked to verify if it matches the description of *b1*. The appending of the navigation action *move* (robot,*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 of Algorithm 4.2 begins by searching the type of the expected object *obj*, which comes from the semantic action model. *obj* is derived from the Add part effects of the selected action. Following this, the process continues by asking the SKB about the super classes of the expected object *obj* step (classes = SKB[type(obj)]). Only the direct 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 example, if *r2* is confirmed to be a room and a kitchen, then only the kitchen class is considered.

Algorithm 4.2

The pseudo code of the semantic knowledge based plan generation process

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

In step (temp-obj = Evidence[obj]), the evidence database is asked to return the expected object which is given a temporary name by this process. The evidence database is then queried in terms of the related properties and relations according to the other observed objects of the expected object step (α = Evidence [properties-and-relations(temp-obj)]).

In step (create- instance-of (temp-obj , α)), the evidential information relating to the expected object is used to construct a temporary instance in the 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 skpg:Room
      ?ch1 rdf:type skpg:Chair
      ?b1 rdf:type skpg:Bed
       ?temp skpg:isContained ?ch1
       ?temp skpg:isContained ?b1
```

}

where *temp* is a temporary symbol used to refer to the current room (where the robot is actually located), i.e. the planning-time object. SKB classifies the newly created instance based on the properties and the relations to the other observed objects, i.e. the chair *ch1* and the bed *b1*. Once the SKB has completed the classification of the planning-time object, the planning process sends another query to the SKB to verify whether or not the classification is consistent with the asserted classes of the expected object *obj* step (if ∀ c ∈ classes: is-instance-of(temp-obj , c)). In step (else if ∃ c ∈ classes: is-not-instance-of(temp-obj, c)), the planning process establishes whether the available observed information reveals a violation of one of the constraints in the definition of the expected object's classes. For the previous example, this is performed by sending the following two queries to the SKB, with the second query only asked when the answer to the first one is "NO".:

ASK { ? *temp* rdfs:subClassOf *BedRoom* }
ASK NOT EXISTS { ? *temp* rdfs:subClassOf *BedRoom* }

4.6 Experiments

In this section, two main testing environments are used to validate the methods presented earlier in the chapter. The first environment is based on analysis of the generated plans in scenario of planning under deterministic conditions with exact plans.

The second testing environment method is to analyse the performance of the planning system by using metrics for performance analysis (such as true positive rate etc.). This

analysis tool supports the investigation of the proposed planning system's behaviour. Navigation and manipulation actions are considered in this scenario.

4.6.1 Environment Setup for Plan Analysis

In this environment, the experiments were conducted according to the programming languages and software available to model the robot environment. The SAMs and SKB were built using Protege[3]. The knowledge base is integrated with a Pellet inference system (Sirin et al. 2007) which is used for reasoning purposes. The task planner was implemented using the Metric-FF (Hoffmann and Nebel 2001) and LPG-td (Gerevini et al. 2004) planners. The algorithm was applied using the Java programming language which is interfaced with Pellet, OWL, and the planners through *Java Application Programming Interface (Java API)*. The ontology of the knowledge base was coded by hand, whereas the planning domain and problem were created automatically by Algorithm 4.1.

Figure 4.1 shows the robot environment which is used to test the benefits of employing a semantic action model along with a knowledge base to support the robot task planner. This environment is divided into eight rooms, namely the Kitchen, Office, BedRoom, BathRoom, LivingRoom, Library, DiningRoom and UtilityRoom. Each room is divided into places, which are represented as small squares, to represent the possible locations that can be used by the robot to navigate around its current room or to go out toward other rooms. This environment also contains some objects related to each room, such as a microwave, table, TV, etc. The ontology representation of this environment is shown

[3] http://protege.stanford.edu

in Figure 4.2. The following conventions are used in these scenarios: *r* denotes robot; p_i denotes position *i*.

4.6.1.1 Knowledge Based Deterministic Planning (Exact Plan)

The utility of the framework proposed in this chapter is demonstrated by explaining some useful scenarios in different situations. There are two scenarios used to verify how the robot task planner is supported by SAMs and SKB. The first of these is a navigation scenario, while the second is a manipulation scenario.

1. **Navigation Scenario**

   Initially, according to Figure 4.1, the robot was placed in the *BedRoom* in position 1, and the user gave an order to the robot to go to the *DiningRoom* at position 12. Assume that, within *BedRoom*, a bed object is at position 4 and there are two doors at positions 5 and 10. There are two scenarios that are used to test the framework in this environment. The first one explains the navigation without using the semantic knowledge base, whereas the second one explains the using of a semantic knowledge base in the navigation.

   i. **Navigation without Knowledge Base**

      The planning domain of this scenario contains the following definition of the move action, which is gained by feeding the SAM shown in Figure 4.4 into Algorithm 4.1 .

      (:action move :parameters (?r-robot ?from-place ?to-place)

      :precondition (and (robot ?r-robot) (place ?from-place)(place?to-lace)

      (at?r-robot ?from-place) (isconnectedto ?from-place ?to-place))

:effect (and (at?r-robot?to-place) (not (at?r-robot?from-place)))

The generated plan contains the following sequence of actions:

move(*r*,*p1*,*p4*),*move*(*r*,*p4*,*p5*),*move*(*r*,*p5*,*p8*), *move*(*r*,*p8*,*p10*), *move*(*r*,*p10*,*p11*), *move*(*r*,*p11*,*p12*)

It is clear that the robot will make contact with the bed in position 4 as the robot does not take into consideration the fact that the bed is an obstacle that must be avoided. This situation may hence lead to the robot sustaining damage.

ii. **Semantic Knowledge Based Navigation Planning**

The planning domain in this scenario is the same as above, but with the addition of a new obstacle definition:

(or(not(obstacle ?to-place)))

The plan, according to this domain, is described below. It is clear that the robot will avoid hitting the bed by moving to position 2 and then position 5, before continuing on its path to position 12. This is evidence that when a robot is aware of the semantics of a situation it can avoid critical situations. According to this modification the generated plan is:

move(*r*,*p1*,*p2*), *move*(*r*,*p2*,*p5*), *move*(*r*,*p5*,*p8*), *move*(*r*,*p8*,*p10*), *move*(*r*,*p10*,*p11*), *move*(*r*,*p11*,*p12*)

2. **Manipulation Scenario**

In this scenario, the environment includes (in addition to the environment described above) some additional objects such as a milk box, table, etc. The user ordered the robot to bring the milk box from the kitchen to the living room and put it in any suitable place. This scenario follows the same steps as in the previous scenario to generate the planning domain, so only the generated plan will be displayed here.

i. Manipulation Planning without Knowledge base

The generated plan is:

take(r,milkbox,p28), *move*(*r,p28,p29*), *move*(*r,p29,p32*), *move*(*r,p32,p34*), *move*(*r,p34,p37*), *move*(*r,p37,p40*), *drop*(*r, milkbox,tv*)

It is clear that the robot could put the milk box on the TV because the planner does not take into account the fact that objects must be put on a flat surface. This information can be gleaned only from semantic knowledge.

ii. Manipulation Planning with Knowledge base

In this situation, the planner gets support from the knowledge base and it defines the object workspaces as any object that has a flat surface. The new generated plan is therefore:

take(r,milkbox,p28), *move*(*r,p28,p29*), *move*(*r,p29,p32*), *move*(*r,p32,p34*), *move*(*r,p34,p37*), *move*(*r,p37,p36*), *drop*(*r, milkbox,table*)

4.6.2 Testing Planning System Efficiency Using Performance Metrics

In this environment the experiments consisted of two scenarios: a manipulation scenario and a navigation scenario. The simulated environment was run under the Robot Operating System (ROS) (Quigley et al. 2009). This ROS is a set of software libraries and tools that help to build robot applications.

4.6.2.1 Metrics for Performance Evaluation

In order to recover from the issue of a lack of benchmark systems in generating symbolic plans under deterministic conditions the evaluation is based on the metrics of false positive rate (FPR) and true positive rate (TPR). Both metrics assume a binary classifier that tries to classify a set of instances as either positive or negative.

$$FPR = \frac{\text{The number of negative instances that are erroneously classified as positive}}{\text{The total number of actual negative instances}} = \frac{\text{FP}}{\text{N}} \tag{4.4}$$

$$TPR = \frac{\text{The number of positive instances that are correctly classified as positive}}{\text{The total number of actual positive instances}} = \frac{\text{TP}}{\text{P}} \tag{4.5}$$

4.6.2.2 Analysing Planning System Behaviour using Performance Metrics

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

consisted of generating the high-level action *take(robot,obj,place)* to pick the object *obj* that was in the fridge by use of a robot arm. The high-level definition of the *take* action will be given in Table 7.1.

The checking process was called once the planner reported that it had succeeded in performing the *take* action. The robot was assumed that it equipped with a camera used to acquire perception information about the picked-up object. This information was then stored in the evidence database.

The *take* experiment was run such that *obj* was asserted to be one of 4 types: *cup*, *bowl*, *glasswater*, and *milkbox*. For each type, the experiment was run 20 times, giving a total of 80 runs. In each run, the type of the object was correctly selected from the 4 available types depending on its description in the problem definition.

Similarly, the navigation scenario consists of the action *move(robot,from,to)* to move the robot from a room identified by *from* to another room identified by *to*. The type was asserted to be one of the eight available room types, i.e. *BedRoom*, *LivingRoom*, etc. Each room type was considered 20 times, resulting in 160 runs in total. For each run, the type of the final location of the robot was selected successfully from the 8 available types.

The problem definition contains objects that are consistent with the actual location, for example the oven object is placed in the kitchen room. The selection of the number of objects related to each room depends on the parameter *Th*. In the case of navigation, the *Th* represents the allowed number of objects in each room. In the case of manipulation, *Th* represents the allowed number of objects which are related to the object in the manipulation action, e.g. the handle object is related to the cup object. In these

experiments, the selected values of *Th* are: 3, 5 and 7 objects. These objects are added to the evidence database while other objects, if available, are hidden.

Table 4.1 shows the results obtained for the three different values of *Th*. The rows of the table represent the grounding process of action parameters, i.e., the replacement of action parameters with suitable value from task planner work space. The first row of each scenario represents the positive grounding result, i.e. the runs where the planning time outcome of the action is the same as the expected outcome. The second row represents a negative grounding result, i.e. runs where the planning time outcome of the action is different from the expected outcome. The columns represent the results of checking the implicit expectations of the expected object (grounded object), so the results are either Matched (M), Unmatched (U), or Ambiguous (A).

Table 4.1: Results from Running the Planning System under Deterministic Conditions for the Actions *take* and *move*. The Cells Represent Number of Runs that result in Matched (M), Unmatched (U) or Ambiguous (A) Outcomes.

		Th = 3objects			*Th* = 5objects			*Th* = 7objects		
		M	**U**	**A**	**M**	**U**	**A**	**M**	**U**	**A**
Navigation	**M**	6	0	20	8	0	20	12	0	20
	U	0	110	24	0	112	20	0	116	12
Manipulation	**M**	2	0	16	6	0	14	10	0	12
	U	0	12	50	0	18	42	0	21	37

In these results, the deterministic planning is able to show a high number of true positives (positive instances that are correctly classified) and zero instances of false positives (negative instances that are erroneously classified as positive). The true

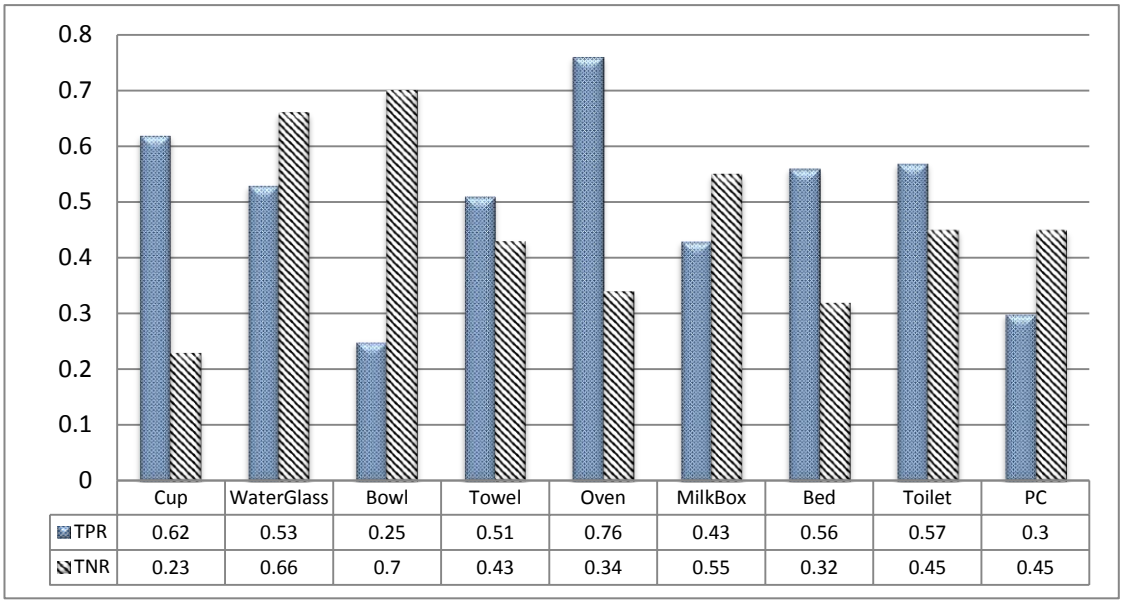
negatives (failure situations) are detected at different rates for the navigation and manipulation scenarios. The values are high in navigation (82%, 85%, 90%) and small in manipulation (19%, 30%, 36%).

The difference in failure detection rates between these scenarios comes from the differences in class definition. In the case of manipulation, a small number of restrictions are used to define each class in the scenario, so a large number of conditions will be treated as negative evidence, and hence ambiguous results will be high. In case of navigation, the classes in the scenario are highly constrained, so fewer numbers of conditions are treated as negative evidence and ambiguous results are low.

The details of the true positive rate (TPR) and true negative rate (TNR) for the different types of the expected object are shown in Figure 4.6 and Figure 4.7. For the manipulation actions, all the successful cases were from plans where the expected object to manipulate was a cup or an oven. For the navigation actions, all the successful cases were from plans where the robot planner successfully generated a move action into either the kitchen or the hall. This is due to the fact that the planning system could use objects that were defined to be specifically related to those types of rooms and objects. For instance, relating a bed to a room was sufficient to conclude that it is a bedroom, while relating a handle to the object (container) in the *take* action caused that object to be classified as a cup.

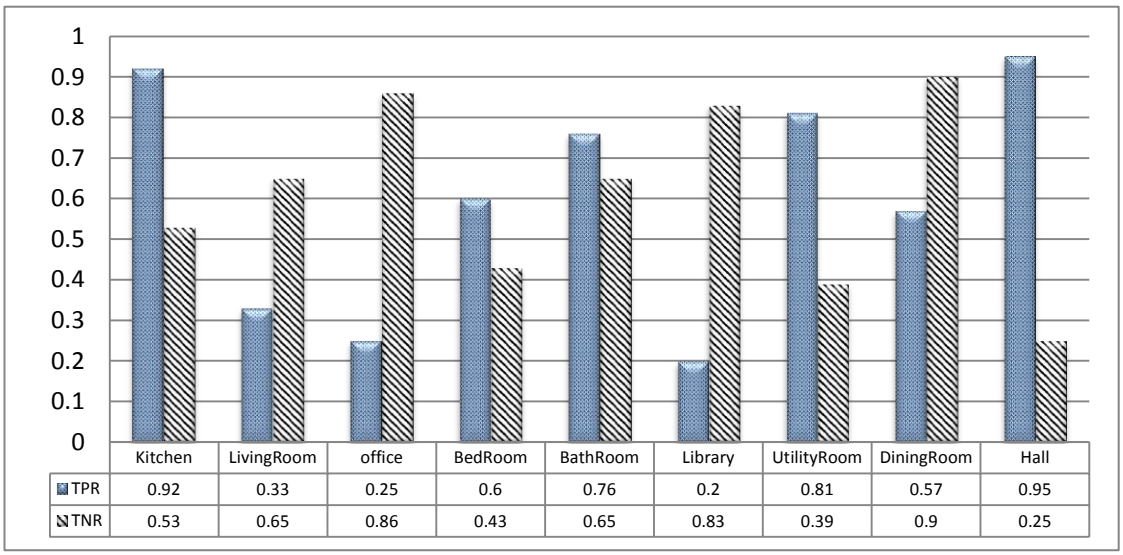
On the other hand, the proper detection of failure situations is also important, as it saves the robot from executing incorrect plans. The true negative rate (TNR) is therefore considered to explain the rate of failure detections, and its value is not zero. This leads to the observation that, if SKB contains restrictions specifically identifying classes of

objects, the planning system can properly generate plans with lower rates of failure cases.



	Cup	WaterGlass	Bowl	Towel	Oven	MilkBox	Bed	Toilet	PC
TPR	0.62	0.53	0.25	0.51	0.76	0.43	0.56	0.57	0.3
TNR	0.23	0.66	0.7	0.43	0.34	0.55	0.32	0.45	0.45

Figure 4.6: True Positive Rate (TPR) and True Negative Rate (TNR) Achieved by Planning System for Different Types of Objects.



	Kitchen	LivingRoom	office	BedRoom	BathRoom	Library	UtilityRoom	DiningRoom	Hall
TPR	0.92	0.33	0.25	0.6	0.76	0.2	0.81	0.57	0.95
TNR	0.53	0.65	0.86	0.43	0.65	0.83	0.39	0.9	0.25

Figure 4.7: True Positive Rate (TPR) and True Negative Rate (TNR) Achieved by Planning System for Different Types of Rooms.

Table 4.2 displays the true positive rate (TPR) and false positive rate (FPR) for the navigation and manipulation scenarios obtained by taking two different cases and

dealing with the ambiguous results. In the first case, the planning system takes the open world approach and considers the ambiguous results as matched because there is no negative evidence in these situations. In the second case, the planning system takes the closed world approach and considers the ambiguous results as unmatched because for some conditions it is not specifically determined whether they are hold or not.

Table 4.2: The Percentage (%) Rates of True Positives (TPR) and False Positives (FPR) of Planning System under Deterministic Conditions for the Actions *take* and *move*. Two World Features are Considered: Open World Treating Ambiguous as a Successful Case and Closed World Treating Ambiguous as a New Case.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
	World Feature	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
Navigation	**Open**	100	17.91	100	15.15	100	9.38
	Closed	23.08	0	28.57	0	37.5	0
Manipulation	**Open**	100	80.65	100	70	100	63.79
	Closed	11.11	0	30	0	45.46	0

The open world approach detects 100% of successful planning cases, but at the same time it suffers from higher rates of false positives in the case of the manipulation scenario.

Both open and closed worlds give a good performance with a low number of objects *Th*. This is because, in closed world approach, the planning system achieves 0% false positives, i.e. 100% specificity and sensitivity (TPR) never equal zero. When using an open world approach this gives 100% of true positives and an FPR less than TPR. Therefore, it can be concluded that the proposed planning system is successful in diagnosing incorrect plans, but it is less good at diagnosing correct plans.

4.6.3 Statistical Analysis of the T-Test

To check the statistical significance of the result generated by the planning system, a T-Test is performed. The T-Test checks the relationship between two variables, in this case two different sets of parameters of the same planning system and it tries to answer the following two questions:

1. **What is the probability that a relationship exists?**
2. **If it does, how strong is the relationship?**

T-test requires a null hypothesis or an expectation to test against. In this case, for the planning system, the null hypothesis is that there is no difference in the performance of the planning system when generating the plans depending on explicit and implicit expectations.

The T-Test assesses whether the means of two groups are statistically different from each other by providing an alpha (α) parameter. If alpha (α) is less than 0.05 then there is a significant difference in the group means.

According to Table 4.1, for each scenario, there are 3 cases that are tested by T-Test depending on the value of the variable *Th*. The American Psychological Association (APA) style (American Psychological Association 2005; Fisher 2015) is used to report about the result of the T-Test. The T-Test in APA style is represented as: "*t*(degrees of freedom) = value, *p* = value". In addition the mean (*M*) and the standard deviation (Std_{dev}) are reported. Where:

Th: represents the allowed number of objects which are related to the places or object.

degrees of freedom: is the number of values in the final calculation of a statistic that are free to vary.

p: is used to determine statistical significance in a hypothesis test.

M: is the average of the numbers (results)

Std_{dev}: is a measure of how spreads out numbers (results) are.

i. <u>Navigation scenario</u>

- *Th* = 3 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 3.25, Std_{dev} = 2.12) and implicit (M = 0.75, Std_{dev} = 0.89) expectations, $t(14) = 3.08$, $p = 0.01$ ($p < 0.05$).
- *Th* = 5 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 3.50, Std_{dev} = 2.07) and implicit (M = 1.00, Std_{dev} = 1.07) expectations, $t(14) = 3.03$, $p = 0.01$ ($p < 0.05$).
- *Th* = 7 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 4.00, Std_{dev} = 2.83) and implicit (M = 1.50, Std_{dev} = 1.31) expectations, $t(14) = 2.27$, $p = 0.04$ ($p < 0.05$).

These results suggest that implicit expectations really do have an effect on the planning system. Specifically, these results suggest that when the planning system uses semantic knowledge base (the source of implicit expectations), the planning system performance increases in navigation scenario.

ii. <u>Manipulation scenario</u>

- *Th* = 3 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 4.50, Std_{dev} = 2.52) and implicit (M = 0.50, Std_{dev} = 1.00) expectations, $t(6) = 2.95$, $p = 0.03$ ($p < 0.05$).

- *Th* = 5 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 5.00, Std_{dev} = 2.00) and implicit (M = 1.25, Std_{dev} = 0.50) expectations, $t(6) = 3.64$, $p = 0.01$ ($p < 0.05$).
- *Th* = 7 shows that, there was a significant difference in the reported performance level of results which obtained from the planning system to generate the plans depending on explicit (M = 5.25, Std_{dev} = 2.99) and implicit (M = 1.25, Std_{dev} = 0.50) expectations, $t(6) = 2.64$, $p = 0.04$ ($p < 0.05$).

These results suggest that implicit expectations really do have an effect on the planning system. Specifically, these results suggest that when the planning system uses semantic knowledge base (the source of implicit expectations), the planning system performance increases in manipulation scenario.

These values indicate that the results obtained by the planning system by depending on the explicit and implicit expectations are significantly (statistically) different with a confidence level 95%.

4.7 Discussion

This chapter has presented a novel high-level approach for generating symbolic plans for robots working in indoor environments. The novelty of the approach comes from using the semantic knowledge domain to calculate implicit expectations that are to be checked and verified when plan actions are successfully generated. These implicit expectations are then checked during the planning time.

Although semantic knowledge plays an important role in many application areas, such as the semantic web (Berners-Lee et al. 2001), its use is still uncommon in mobile robotics. Notable exceptions include applications for creating a knowledge processing

system (Tenorth and Beetz 2013b), for classifying map spaces for navigation tasks (Galindo et al. 2005), and for publishing and sharing knowledge in multi-robot environments (Chella et al. 2002). The process of symbolic plan generation needs to be implemented at different layers, from creating the planning domain and problem definitions to high-level planning. This means that semantic knowledge can contribute to a multi layered planning process.

The use of semantic knowledge in task planning aids the diagnosis of incorrect plans that cannot be diagnosed by traditional planning approaches. The reason for this is that planning, in many situations, depends only on the explicit effects of actions. This is not sufficient for diagnosing unexpected situations. A direct consequence of using semantic knowledge in creating plans is that the process of planning to achieve tasks becomes less computationally demanding. This is due to the task planner not reasoning about the details of the objects manipulated by the planning domain actions.

The open world and closed world features are used to provide the planning system with two abilities to deal with the ambiguous results that appear in table 4.1. In the case of open world feature, the ambiguous results are treated as matched (success) cases. This leads to high percentage of TPR (100%) which means high rate of correct classification, in the same world feature the rate of FPR is not zero which means that some negative results are classified erroneously as positive.

While in the case of closed world feature, the ambiguous results are treated as new cases. This leads to low percentage of TPR which means low rate of correct classification, in the same world feature the rate of FPR is zero which means that all negative results are not classified erroneously as positive.

Although the proposed approach helps in developing more robust plan generation, it still suffers from the failure to handle uncertainty in world states as well as actions with more than one possible outcome. In the next chapter, another approach of semantic knowledge based plan generation will be presented that aims to handle uncertainty in action details and world states. The goal is to develop an approach that can be applied to different world scenarios. The quality of the ontology of robot environment is not tested separately, but the overall performance of the planning system (including ontology) was evaluated by using the performance evaluation metrics such as FTP and TPR. It is a good idea to test the robot ontology separately before adding it into the planning system.

The main contributions of this chapter are:

- The robot knowledge base is modelled as ontology which represents objects, places and the relations between each of them. This ontology becomes the source of information necessary to support the robot task planner in generating plans for robot tasks.
- The robot high-level actions are described in a new model called a Semantic Action Model (SAM). SAMs consists of two parts: the profile part and the process part. Then the variables for the action components, i.e. inputs, outputs, preconditions and effects, are related to semantic knowledge base on the semantics of the action parameters.
- The robot knowledge base is integrated with a Semantic Action Model (SAM) to generate a planning domain that will be used to support the task planner and thereby improve the robot efficiency. This integration makes the robot more aware of the facts and information that an ordinary human is expected to know.

- A new algorithm is created to engage the information in an action model (SAM) with its relevant data in the knowledge base, and to produce a planning domain represented in PDDL format. The planning domain is then fed into an external planner to produce the plans for robot tasks.
- Then, a general purpose planning algorithm has also been defined, which can support planning under deterministic conditions, and is based on using ontology to represent SKB.

4.8 Summary

This chapter represented the robot semantic knowledge as ontology which is stored in the OWL format. This representation endows the planning system with reasoning capabilities to infer new information to support the task planner in generating symbolic plans for robot tasks. This chapter also presented a new model representation of high-level robot actions. This type of model is called a semantic action model (SAM). This representation describes action components (parameters, preconditions and effects) as ontology which is represented in OWL style. It facilitates the integration of action components with the knowledge base. This integration provides the robot with the ability to be more aware of the semantics of the places and objects in its environment. This awareness allows the robot to avoid potential pitfalls such as hitting obstacles, losing its way to other rooms, and putting objects in wrong workspaces. It also allows the robot to detect whether or not it is possible to carry objects. These features make the robot more flexible in completing its task. The chapter then presented a novel high-level approach for generating symbolic plans for robot tasks in indoor environments.

Chapter 5

Semantic Based Planning under Probabilistic Conditions

5.1 Introduction

Generating plans in real world environments using a mobile robot planner is a challenging task due to the large amount of information, uncertainty and dynamics in the robot environment. With this in mind, task planning should take these issues into consideration when generating plans. The ability of a robot to reliably generate task plans and detect failures is essential to its performance and autonomy.

The semantic knowledge domain has been proposed as a source of information for deriving implicit information and generating semantic plans. In this chapter, a method is proposed and used to increase the reliability of generating symbolic plans. This method is to extend the semantic knowledge base (SKB) plan generation to take into account the amount of information and uncertainty related to (i) existing objects, (ii) their types and properties, and (iii) their relationships with each other. This approach constructs plans by depending on probabilistic values which are derived from statistical relational learned models such as Markov Logic Networks (MLN).

An MLN module will be established for probabilistic learning and inference and combined with semantic information to provide a basis for plausible learning and reasoning services in support of robot task planning. The MLN module is created by using an algorithm to transform the knowledge stored in SKB into domains, predicates and formulas. These represent the main building blocks of the module, which is one of the main contributions of this chapter.

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. These expectations are matched against MLN answers. MLN answers are based on evidence obtained from robot perceptions and fed into the trained MLN model during the inference phase. By providing a means of modelling uncertainty in planning system architecture, task planning serves as a supporting tool for robotic applications that can benefit from probabilistic inference within a semantic domain. This approach is illustrated using test scenarios run in a domestic environment using ROS.

5.2 Overview of the Approach

Plan generation by an autonomous robot planner acting in human environments is a complex task. It involves dealing with uncertainty and nondeterministic situations in which 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 sometimes cause failure in a robot’s operation.

To deal with unexpected planning contingencies, robotic task planning employs probabilistic inference procedures, based on the reasoning techniques or learned models, to ensure the generation of its plans does not deviate from their intended course of action (Bouguerra et al. 2007a). Most approaches to plan generation focus on: (i) deterministic information about the robot environment, i.e. exact objects and their properties, for instance “in the kitchen, there is a fridge and an oven”; and (ii) explicit actions' details (preconditions and effects), for example the explicit effect of grasping a glass of water would be that the robot is holding a glass of water.

In a real world environment this is not always realistic as planning with uncertainty is a complex process. Therefore, more advanced forms of probabilistic reasoning should be proposed which engage with semantic knowledge to derive probabilistic implicit information about the existence of objects and their types. For example, a robot moving into a living room could be expected to see at least a tv-set and a sofa. If the robot is entering a kitchen instead, it should have more probability of seeing a fridge and a sink. These probabilities are details that would add more complexity to the task planner, if the task planner has been left to reason about them. Therefore, it is important to build a separate unit from a semantic knowledge base and the robot action description. This unit would have the ability to learn from a knowledge base (SKB at object level) and then infer probable information about the robot environment from the evidence database to support robot task planning.

In this chapter, the robot semantic knowledge base is represented with Description Logic (DL) (Lukasiewicz 2008), which has the ability to infer the types of things (objects and places) and the automatic classification of things based on their classes and properties. This approach enables the robot to derive new statements from its existing knowledge. Pure description logic inference is completely deterministic, so it is often desirable as a way of representing uncertain information in a way that can be more useful to the robot planner.

Markov Logic Networks allow the drawing of probabilistic inferences that combine the expressiveness of first-order logics with the representation of uncertainty (Richardson and Domingos 2006). Such relational structures which include variable sets of objects can have an influence on the propositions relevant to the robot's tasks. Therefore, they cannot be accounted for in a model that involves a fixed set of propositions. What is

required is unification, within a single representation formalism, of the principles of: (i) first-order logic, which makes objects and relations the main building blocks of the representation; and (ii) probabilistic graphical models, which enable reasoning in the face of uncertainty. This approach enables, by combining the respective semantics, a language that possesses a sufficient level of expressiveness for robots to be equipped with the much-needed capability of reasoning about situations as they arise in real-world applications.

The main contributions of this chapter are (i) the construction of an MLN from the semantic knowledge base to support the robot task planner in situations of uncertain object existence and object type, (ii) an algorithm for creating an MLN model from the semantic knowledge base and (iii) the development of a probabilistic planning approach which is able to take into account uncertainty in world states, action effects, and the way that expectations are interpreted in the semantic domain knowledge. Thus a probabilistic quantitative model of uncertainty, such that actions are allowed to have different outcomes, each with a probability of occurrence, can cause planning to be unreliable. Using probabilities make it possible to go beyond a Boolean treatment of whether an expectation is verified.

The MLN template has the ability to learn from training data which is extracted from the knowledge base. The trained model is then ready to answer queries concerning information uncertainty that are issued during the planning process. These queries are issued from the planning system when the knowledge base has no answer about uncertain information, because it cannot handle uncertainties (this was the limitation of the approach presented in Chapter 4). The answer describes the state of the objects (places or entities) by predicting their place of existence and type.

Chapter 4 showed how semantic domain knowledge can be used in the process of generating symbolic robot plans. The key idea is to use such knowledge to compute implicit expectations of the objects that can be recorded at planning time by the robot. This ensures the plans are constructed correctly. The approach addressed actions with deterministic effects (i.e. having only one outcome), and world states. The result of evaluating the implicit expectations was treated in a Boolean setting, that is either true, false, or unknown. These limitations arise predominantly due to the fact that the process does not represent inherent uncertainty of world states.

As uncertainty is an ever-present feature in mobile robotics, it has an impact on the actions of robots as well as their description of their environment. In the presence of uncertainty, even the best prepared plans can fail. In particular, the task planning process can combine different evidence in a systematic way. Therefore, given the priori probabilities of the possible action outcomes, the available semantic knowledge, and the evidences the task planner is able to estimate the probability of whether a certain expectation is verified. An output example may be: “the robot is in a kitchen with 0.9 probability”. In the framework proposed in Chapter 4, that example would have resulted in a planning fail. Moreover, the fact that a probability can be estimated for each outcome of an action enables a more informed decision about how to proceed (plan generation successful, plan generation failed, or more information needed) than with just a Boolean approach.

5.2.1 Motivation

Chapter 4 discussed the effects of uncertainty on plan generation. The plan generation can be facilitated by supporting the robot task planning with techniques that give it the ability to reason about uncertain cases. Therefore, robot planning systems which include statistical relational models allow their plans to represent different situations adequately. As those plans are generated off-line, the effects of their actions need to be estimated at planning time, since the actual effects are not known beforehand. To estimate the actual effects of actions effectively, the planning process also needs to reason about the uncertainty inherent in the action outcomes and world states.

This section describes a different plan generation process that is able to reason about uncertainty. A model is developed that takes into account quantitative uncertainty in the form of possible world probabilities, 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 the evidence base and action effects . As a result of using probabilities, it is possible to go beyond a simple Boolean treatment of whether an expectation is verified. In other words, a probability for whether a certain expectation of the world state is verified during the planning process. An example would be “the robot is in the bedroom with probability of 0.90” instead of returning an ambiguous or false planning result. Moreover, the fact that an estimated probability of each possible world state can be support a more informed decision about how to proceed by considering the plan generated as successful, failed, or lack of information. Indeed, this is the main advantage of probabilistic planning approach over deterministic

planning approach as it uses 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 finding a bed is zero, while the probability of finding one 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), there is unfortunately no available DL system that supports probabilities. Thus, in this work, 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 (evidence base) reveals:

• Whether or not an object that exists in the real world can be 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.

• Ways to classify objects by depending on their associated properties such as their place, function or position.

A semantic knowledge base is proposed to support the robot planning system by handling the issues of (i) uncertainty regarding the existence of objects in a certain place or (ii) predicting types of objects or places, as well as the relationships between them. For instance, to insert an action such as *move(robot,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's location, it is necessary for the planner to obtain 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 in MLN 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 and Domingos 2006), directed graphical models (or 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 (MLN) are used in this thesis as a statistical relational model to support the task planning process.

The planning process uses the prior probability distribution over the possible spectrum of the world states, which are stored in the trained MLN as formula weights, together with the semantic knowledge base and the observed model to compute the state of the

possible world probability. The method brings with it constraints over atomic classes of observable objects, and it is easy to add constraints over values of other attributes, e.g. color ∈ {red, yellow, white}.

Example: Assume that the planning system needs to append the navigation action *move*(*robot*,*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 explain the observed beds and sinks in the current location.

5.3 Probabilistic Planning System Architecture

Figure 5.1 represents the probabilistic planning system architecture proposed in this chapter. Two new units are added in this framework: the 'SKB to MLN' Algorithm, and the MLN module. Other components have the functionalities which were explained in Chapter 4 (Section 4.4). The following sections describe the new components and how they are used in this chapter

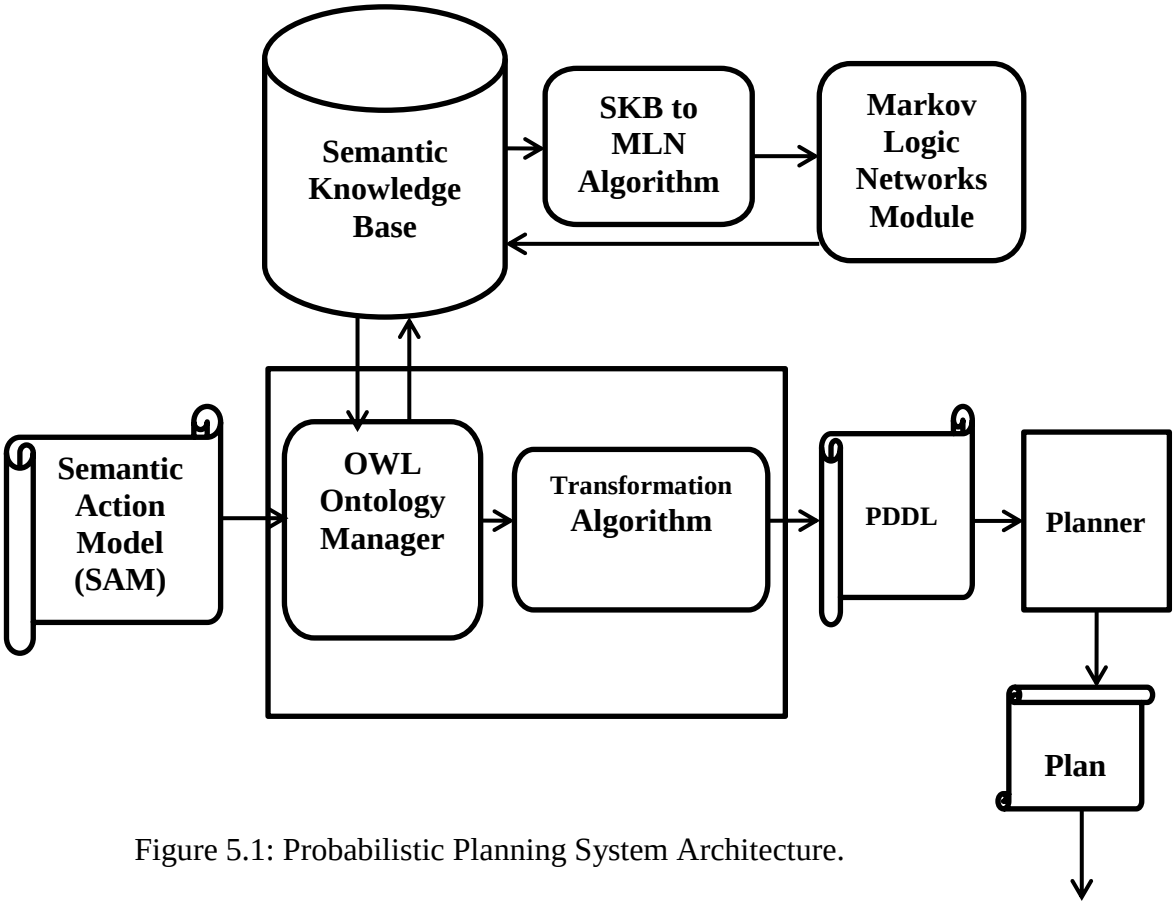


Figure 5.1: Probabilistic Planning System Architecture.

5.3.1 Markov Logic Networks (MLNs)

The formal definition of an MLN L is given as a set of pairs $<F_i , w_i>$, where F_i is a formula in first-order logic and w_i 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. More details are given in (Jain 2011a):

* X is a set of Boolean variables. Each possible grounding for every predicate appearing in L adds a Boolean variable (ground atom) to X.

* G is a set of weighted ground formulas, i.e. a set of pairs $<F_j', w_j'>$, where F_j' is a ground formula and w_j' 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:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|L|} w_i n_i(x) \right) = \frac{1}{Z} \exp\left(\sum_{j=1}^{|G|} w'_j f'_j(x) \right) \quad (5.1)$$

$$Z = \sum_{x' \in X} \exp\left(\sum_i w_i n_i(x') \right) = \sum_{x' \in X} \exp\left(\sum_j w'_j f'_j(x') \right) \quad (5.2)$$

where Z is a normalisation constant and $n_i(x')$ denotes the number of true groundings of F_i in x. There are two phases of MLN working here: learning and inference.

5.3.2 MLN Learning Phase

The learning of a statistical relational model involves the construction of a model from observed training data. The structure of the model can be known a priori, leaving only the parameters to be determined, as shown by Kok and Domingos (2005). It can also 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 and Domingos (2005). 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 an MLN, 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. As such 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 $X = x$.

In the case of the planning system which is proposed in this chapter, the MLN will be created from the SKB, in particular from the class level in the ontology. The training database will be created from objects in the SKB, in particular from object level in the ontology. The different types of component which are involved in the learning phase are shown in Figure 5.2.

There are several different types of learning methods that can be used to train MLN models. In this study, two types of learning method are used, as discussed in the next paragraphs.

Maximum Pseudo-Likelihood Learning: This method operates by maximising the Pseudo-Likelihood:

$$P^*(X = x) = \prod_{k=1}^{|x|} P(X_k = x_k \mid MB_x(X_k)) \tag{5.3}$$

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 and Domingos 2006). The Pseudo-Likelihood approximates P(X = x) by making strong independence assumptions, and 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. This means that they can be trained to represent a conditional distribution rather than a full-joint 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 unknown and everything else is given), discriminative training can yield models with superior performance. Methods for the discriminative training of MLNs were introduced by (Singla and Domingos 2005).

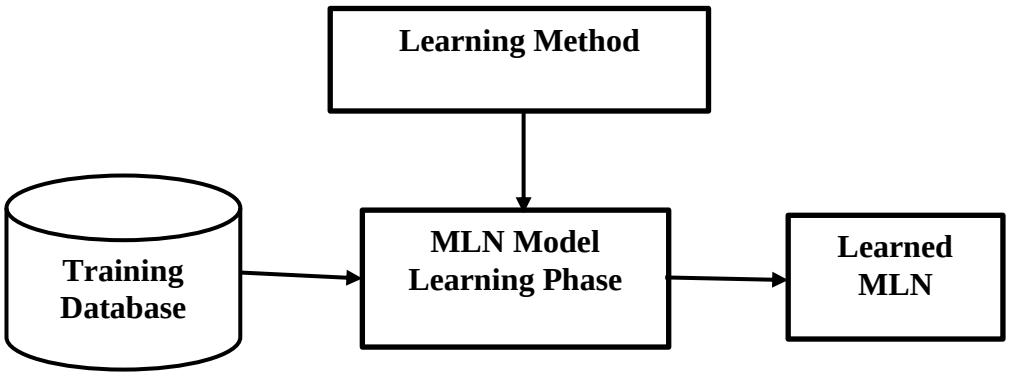


Figure 5.2: MLN Learning Phase

In many applications, a priori is known whereby the predicates will be evidence and ones will be queried. 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,* then the conditional likelihood of Y given X is:

$$P(y|x) = \frac{1}{Z_x} \exp(\sum_{i \in F_Y} w_i n_i(x,y)) = \frac{1}{Z_x} \exp(\sum_{j \in G_Y} w_j \, g_j(x,y)) \tag{5.4}$$

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 thus:

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(y|x) &= n_i(x,y) - \sum_{y'} P_w(y'|x) \, n_i(x,y') \\ &= n_i(x,y) - E_w[n_i(x,y)] \end{aligned} \tag{5.5}$$

5.3.3 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. There are different types of inference methods used to

infer new information from a learned model of an MLN, so in this study two of these methods are used. Figure 5.3 explains the main components which are used in the inference phase (query phase) of the MLN.

Markov chain Monte Carlo (MCMC) (Koller and 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.

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 can soundly handle distributions with deterministic and near-deterministic dependencies, which, given the possibility of logical modelling, abound in Markov logic networks. MC-SAT applies slice sampling to Markov logic, using SampleSAT (Wei et al. 2004) 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:

$$\phi_k(x) = \exp^{w_k f_k(x)} \tag{5.6}$$

which has value e^{wk} if c_k is satisfied, and 1 otherwise. More details are given by Poon and Domingos (2006).

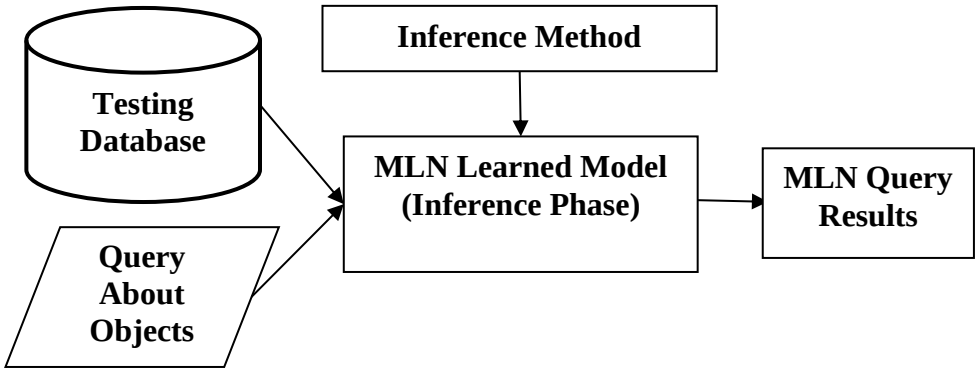


Figure 5.3: MLN Inference (Query) Phase

5.3.4 SKB to MLN Algorithm

Algorithm 5.1 is the proposed algorithm which is used to create an MLN from an SKB as follows:

Algorithm 5.1

```
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
```

5.3.5 Examples

1- Suppose that the planner has to insert the movement action *move(robot, r2, r1)*, where *r2* is a living room and *r1* may be a bedroom or an office. There are therefore two possible values for *r1*. Thus it is assumed that the *r1* value depends on the result of the query that is returned from the trained 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(robot,r2, r1)* with *r1* = office.

2- Continuing the example above, the implicit expectations of being in *r1* are determined based on the type of *r1*. If *r1* 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.

5.4 Testing Scenarios

This section presents experimental simulations to collect data for the purpose of statistically evaluating the performance of the proposed framework. Due to a lack of benchmark systems in generating symbolic plans under probabilistic conditions, the evaluation is based on the metrics of performance, such as TPR, etc. (which were explained in Chapter 4), along with accuracy, precision and recall which will be explained in Section 5.4.2.

The testing scenario involves performing navigation and manipulation tasks in a house environment. The house consists of 9 rooms: *Kitchen*, *Office*, *BedRoom*, *BathRoom*, *Library*, *UtilityRoom*, *LivingRoom*, *Hall* and *DiningRoom*. The block diagram of this environment has been shown in Chapter 4 in Figure 4.1. In these scenarios, the rooms have not yet been identified and the robot depends on the MLN module to identify them. The environment setup for this chapter follows the same setup procedures of Chapter 4 Section 4.6.2.

5.4.1 Simulation Results

In this section, the Probabilistic Cognition for Technical Systems(Jain 2011b) software was used to train the MLN model and to infer new information from it in the experiments. The data of plan generation for manipulation and navigation actions inside a house environment have been collected.

The process of creating an MLN from an SKB depends upon Algorithm 5.1. The interface process between the planner, SKB, SAMs and the MLN unit is achieved via java API and OWLAPI under ROS environment.

5.4.2 Metrics for Performance Evaluation

The same metrics defined in Section 4.6.2.1 for performance evaluation are used in this section. In order to recover from the issue of a lack of benchmark systems in generating symbolic plans under probabilistic conditions the evaluation is based on accuracy, precision and recall metrics in addition to TPR and FPR.

The accuracy shows how close a measured value is to the actual value and represents the proportion of all the instances that are correctly classified, i.e.:

$$accuracy = \frac{TP + TN}{P + N} \tag{5.7}$$

Precision shows how close the measured values are to each other, and represents the proportion of correctly classified positive instances:

$$precision = \frac{TP}{Tp + FP} \tag{5.8}$$

The recall shows how many of the true positives were recalled (returned) and is represents by:

$$recall = \frac{TP}{TP + FN} \tag{5.9}$$

5.4.3 Manipulation Scenario

In the first scenario, the robot will take an object from the fridge and place it on its tray so that it can be carried to another location in 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. For example 'handle is related to cup'. In this environment, there are 18 objects, 4 of which can be picked up.

5.4.4 Navigation Scenario

In this scenario, the robot is acting in a house environment that comprises 8 rooms of different types (kitchen, office, etc.). In each room there are furniture items that are typical for that type of room. For instance, in a kitchen, there is an oven, a sink, etc. In total, there are 18 different types of objects that can exist in any room. The semantic knowledge used in this scenario is more complex than in the previous scenario. The objects' 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.

The task is for the robot to move from one room to another room. The robot can face the situation where it is uncertain about the rooms or the places it has passed, so it is the role of the probabilistic planning system to guide the robot in the right way.

5.4.5 Threshold for Decision Making

The probability of selecting the desired predicates from the trained MLN depends on the standard deviation (Std_{dev}), which is a statistical measure used to quantify the dispersion or variation in a distribution. Std_{dev} is calculated from the weights associated with all the formulas in the MLN model. It is used with the mean values of the formula weights and compared against the values associated with the queried action object in the current plan. This answer is derived from the MLN model in an inference engine phase as depicted in Figure 5.3. There are three types of answer: *matched,* when the probability associated with each predicate is greater than (mean + Std_{dev}); *unmatched,* when the probability associated with each predicate is below (mean- Std_{dev}); lack of information

when the value of the probability that associated with the predicate lays between (mean + Std_{dev}) and (mean- Std_{dev}). There are 3 cases which are considered to obtain the value of (Std_{dev}) and each depends on the variable (*Th*). This variable represents the number of related objects in the room or in relation to other objects. The considered cases are when the number of related objects equals 3, 5 and 7.

5.4.6 Example

This example shows how a trained MLN can be created from a template of MLN by using training data. Firstly, classes' names, roles definitions, training data and MLN are presented. Then, by using test data (evidence database), this example shows how the trained MLN will answer the queries.

The house environment is explained above, and consists of 9 rooms and 18 objects. For simplicity, only 4 of the rooms and 4 of objects are presented here. These rooms and objects have not been identified yet and the robot depends on the MLN module to identify them. The semantic knowledge base contains, among other things, the following class and role definitions:

Classes
Room, Object
Properties:
isContained, its domain is *Room* and its Range is *Object*.
isLocatedAt, its domain is *Object* and its Range is *Room*.
Rules
R1: *Kitchen* is *Room* and *Kitchen isContained KitchenObject*
R2: *Office* is *Room* and *Office isContained OfficeObject*
R3: *BedRoom* is *Room* and *BedRoom isContained BedRoomObject*
R4: *KitchenObject* is *Object* and *KitchenObject isLocatedAt Kitchen*

R5: *OfficeObject* is *Object* and *OfficeObject isLocatedAt Office*

R6: *BedRoomObject* is *Object* and *BedRoomObject isLocatedAt BedRoom*

These definitions are then translated from SKB style into MLN style by using Algorithm 5.1 to create the MLN model:

Domains:

Place = {*kitchen1, office1, bedroom1*}

Object = {*cup1, pc1, bed1, cup2, pc2, bed2*}

Predicates:

Room(*place*), *Object*(*entity*)

isContained(*place*, *entity*), *isLocatedAt*(*entity*, *place*)

Formulas:

Room(x) ^ KitchenObject(y) ^ isLocatedAt(y, x) => Kitchen(x)

Room(x) ^ OfficeObject(y) ^ isLocatedAt(y, x) => Office(x)

Room(x) ^ BedRoomObject (y) ^ isLocatedAt(y, x) => BedRoom (x)

Kitchen(x) ^ Object(y) ^ isContained(x, y) => KitchenObject(y)

Office(x) ^ Object(y) ^ isContained(x, y) => OfficeObject(y)

BedRoom (x) ^ Object(y) ^ isContained(x, y) => BedRoomObject (y)

This model becomes a template, and can be trained by using training data e.g.:

Objects:

Room(k1), Room(o1), Room(b1),

Object(cup3), Object(pc3), BedRoomObject (bed3), KitchenObject(oven1),

KitchenObject (fridge1), KitchenObject (milkbox), OfficeObject(pc3),

OfficeObject (chair1)

Relations:

isContained(k1,oven1), isContained(k1,fridge1), isContained(o1,pc3), isContained(o1,chair1), isContained(k1,milkbox), isLocatedAt(oven1,k1), isLocatedAt(fridge,k1), isLocatedAt(pc3,o1), isLocatedAt(chair1,o1), isLocatedAt(milkbox, k1)

The training process will begin when it receives all the necessary information (as shown in Figure 5.2) and the planning system requests for the process to start. The learning process results in a trained MLN which is ready for any query issued by the planning system. The resulting learned MLN is as follows:

0.564 *Room(x) ^ KitchenObject(y) ^ isLocatedAt(y,x) => Kitchen(x)*
0.325 *Room(x) ^ OfficeObject(y) ^ isLocatedAt(y,x) => Office(x)*
0.129 *Room(x) ^ BedRoomObject (y) ^ isLocatedAt(y, x) => BedRoom (x)*
0.212 *Kitchen(x) ^ Object(y) ^ isContaining(x,y) => KitchenObject(y)*
0.184 *Office(x) ^ Object(y) ^ isContaining(x,y) => OfficeObject(y)*
0.091 *BedRoom (x) ^ Object(y) ^ isContained(x, y) => BedRoomObject (y)*

Next, this learned MLN model is used to predict and infer the probability of (i) an object's existence, (ii) its type and (iii) its relations to other places or objects. For example, consider the robot generates the following perception data (evidence database or testing data):

Room(r1),Room(r2),Room(b2),
KitchenObject*(table),* KitchenObject*(microwave),* KitchenObject*(washingmachine),*
OfficeObject*(chair),* OfficeObject*(sofa),* OfficeObject*(pc),isContained(r1,table),*
isContained(r2,chair),isContained(r2,pc),isContained(r1,microwave),
isContained(r1,washingmachine),isContained(r2,sofa),
isLocatedAt(table,r1),isLocatedAt(chair,r2),isLocatedAt(pc,r2),
isLocatedAt(microwave,r1),isLocatedAt(washingmachine,r1),isLocatedAt(sofa,r2).

The planning system selected the MCMC inference method and queried about the type of room *r1*, which is in the plan and associated with the action *move(robot, r2, r1).* The result is shown in Table 5.1.

Table 5.1: The Result of Querying About the Type of Room r1

No.	Things	Probability
1	Kitchen(r1)	0.791
2	Office(r1)	0.28
3	BedRoom(r1)	0.071

According to the values of mean = 0.25 and standard deviation = 0.158, then *r1* type is determined to be *kitchen*.

5.4.7 Probabilistic Plan Generation

In this section, the same experimental setup as in Chapter 4 (Section 4.6.2) is used to evaluate the performance of the probabilistic semantic knowledge planning system. However, the world states for the planning system have more than one possible state, each with a given probability of occurrence. This is due to the action effects each having two possible outcomes. For the manipulation scenario, i.e. the *take(robot,obj,place)* action, the first world state expresses the possibility of generating this action to take the desired object *obj*, while the other state expresses the possibility of generating the *take* action and taking a different object.

For the navigation scenario, the action *move(robot,from,to)* is used to move the robot from its initial room *from* to its destination room *to*. The first state reflects the situation where the planning system generates the *move* action but the robot stays unintentionally in the *from* room. The other state expresses the case where the planning system generates the *move* action and the robot effectively ends up in room *to*. Probabilistic

planning was also provided with an observed data set (evidence database), which specified the probabilities of not observing or missclassifying an object.

The probabilistic planning system was evaluated using three values of standard deviation. Each one was calculated based on the number of objects that related to the rooms or other objects in the robot environment. The value of each Std_{dev} is computed from the weights associated with the formulas in the trained MLN model.

Experiments were run whereby 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, each with their associated probability. 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* outcome represents the incorrect action effect while *out_2* represents the correct action effect. Three probability distributions relate to these outcomes: for the first distribution p(*out_1*) = 0.3, and p(*out_2*) = 0.7, the second distribution p(*out_1*) = 0.7, and p(*out_2*) = 0.3 whereas for the third one p(*out_1*) = 0.5, and p(*out_2*) = 0.5 . For example, consider the action outcome of *move(robot, r1,r2),* designed to move the robot from *LivingRoom r1* to Kitchen *r2*. When p(*out_1*) = 0.3 and p(*out_2*) = 0.7, the effect of the action is: the robot ends up in the kitchen. This effect will be compared with the answer returned from the MLN when the planning system asks it about the type of *r2*. If is the two results match then the planning system will continue to generate the other actions needed 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. If the other outcome also does not match, then the planning system will ask another module such as the

Information Gathering module (which will be explained in Chapter 6) or SRRM (will be explained in Chapter 7) to extend the robot world states with new information.

For each combination of object, room type, and probability distributions of action effects, the experiment was repeated 10 times. Therefore the total number of runs was 1920 for the navigation. While for the manipulation, the experiment was repeated 20 times so the total number of runs was 960.

Table 5.2 and Table 5.3 summarise the results of the probabilistic planning of both types of action, i.e. manipulation and navigation, with open and closed world features. The rows represent the action outcome with the highest probability in the action model, while the columns show the results of the MLN answer to compute the possible world. These were computed by selecting the outcome with the highest weight. Hence there are two possible outputs: **M**atched and **U**nmatched.

The true positive rate (TPR) and the false positive rate (FPR) for probabilistic planning were computed by considering the outcome (as explained above) *out_2* as the positive case and *out_1* as negative case. Table 5.4 shows TPR and FPR for both types of action in the navigation and manipulation scenarios. The results indicate good performance because TPR tends to be high and FPR tends to be low. As in the case of deterministic planning (Section 4.6.2), one can notice that the performance improves when (*Th*) is higher. Similarly, one can notice that the performance in the navigation scenario is much better than that of the manipulation scenario. This is due to the number of constraints and the number of related objects in the environment.

Table 5.2: Results from Running the Planning System under Probabilistic Conditions for the Actions *take* and *move* with Open World Feature. Each Cell Represents the Number of Runs that Result in Matched (M) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results, or Unmatched (U) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
		U	**M**	**U**	**M**	**U**	**M**
Navigation	***out_1***	650	142	686	130	704	112
	out_2	96	1032	72	1032	24	1080
Manipulation	***out_1***	339	117	366	90	396	48
	out_2	168	336	152	352	96	420

Table 5.3: Results from Running the Planning System under Probabilistic Conditions for the Actions *take* and *move* with Closed World Feature. Each Cell Represents the Number of Runs that Result in Matched (M) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results, or Unmatched (U) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
		U	**M**	**U**	**M**	**U**	**M**
Navigation	***out_1***	736	104	760	80	752	64
	out_2	146	934	98	982	74	1030
Manipulation	***out_1***	328	104	352	80	424	32
	out_2	168	360	156	372	84	420

Table 5.4: The Percentage (%) Rates of True Positive Rate (TPR) and False Positive Rate (FPR) of the Planning System under Probabilistic Conditions for the Two Actions move and take. Two World Features are Considered: Open and Closed World Features.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
	World Property	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
Navigation	**Open**	91.49	17.93	93.48	15.93	97.83	13.73
	Closed	86.48	12.38	90.93	9.52	93.3	7.84
Manipulation	**Open**	66.67	25.66	69.84	19.74	81.4	10.81
	Closed	68.18	24.07	70.45	18.52	83.33	7.02

The results of accuracy, precision and recall are given in Table 5.5, Table 5.6 and Table 5.7 respectively. It can be noticed that the probabilistic planning system is highly accurate in detecting successful and failed planning in the navigation scenario, but is less accurate in the manipulation scenario. As with FPR and TPR, accuracy, precision and recall improve when the environment contains a greater number of related objects (*Th* is higher).

Table 5.5: Accuracy of Planning System under Probabilistic Conditions in Navigation and Manipulation Actions Using the Two World Features Open and Close. The results are given as percentages (%).

	World Property	*Th* = 3 Objects **Accuracy**	*Th* = 5 Objects **Accuracy**	*Th* = 7 Objects **Accuracy**
Navigation	Open	87.60	89.48	92.92
	Closed	86.98	90.73	92.81
Manipulation	Open	70.31	74.79	85.00
	Closed	71.67	75.42	87.92

Table 5.6: Precision of Planning System under Probabilistic Conditions in Navigation and Manipulation Actions Using the Two World Features Open and Close. The results are given as percentages (%).

	World Property	*Th* = 3 Objects **Precision**	*Th* = 5 Objects **Precision**	*Th* = 7 Objects **Precision**
Navigation	Open	87.90	88.81	90.60
	Closed	89.98	92.47	94.15
Manipulation	Open	74.17	79.64	89.74
	Closed	77.59	82.30	92.92

Table 5.7: Recall of Planning System under Probabilistic Conditions in Navigation and Manipulation Actions Using the Two World Features Open and Close. The results are given as percentages (%).

		Th = 3 Objects	*Th* = 5 Objects	*Th* = 7 Objects
	World Property	**Recall**	**Recall**	**Recall**
Navigation	Open	91.49	93.48	97.83
	Closed	86.48	90.93	93.30
Manipulation	Open	66.67	69.84	81.40
	Closed	68.18	70.45	83.33

5.4.8 Sabotaged SAMs

Experiments were run to test the effect of incorrect action models on the performance of probabilistic planning. In these experiments, the SAM used by the probabilistic planning was sabotaged. This means that the probabilities assigned to the possible action outcomes were not always correct. In these experiments, the planning system only generated plans for the navigation scenario using the closed world feature. The action model gave a probability of 0.3 to the first outcome and a probability of 0.7 to the second outcome, i.e., p(*out_1*) = 0.3, p(*out_2*) = 0.7.

Experiments were conducted whereby the two rooms assigned to the two possible outcomes of *move(robot,from,to)*, may be any of the available room types (5 rooms involved in this experiment). For each possible collection of types, 10 experiments were run, so that the total number of experiments run was 250. Each time, the destination *to* room of the robot was selected from the robot world state and compared with the result returned from MLN.

In order to analyse the results of the probabilistic planning system with a sabotaged action model (SAM), the same experiments were repeated where the action model was not corrupted. Table 5.8 summarises the results achieved by the planning process when both action models are used, i.e. sabotaged vs non-sabotaged models. The corresponding rates of false positives (FPR) and true positives (TPR) are given in Table 5.9.

The results explain the performance of the planning system, in terms of FPR and TPR, by a comparison between the sabotaged action model and the accurate action model. The planning system with a sabotaged action model is less accurate and less precise than planning system with an accurate action model. Yet, the difference in accuracy and precision is small, as shown in Table 5.10. It can be concluded that semantic knowledge supports robot task planning to reliably generate plans even when the SAM model is sabotaged.

Table 5.8: The Results from Running the Planning System under Probabilistic Conditions for Navigation Actions with Accurate and Sabotaged SAM Models. Each Cell Represents the Number of Runs that Result in Matched (M) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results, or Unmatched (U) between Action Outcome (*out_1* or *out_2*) and Probabilistic Results.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
		U	**M**	**U**	**M**	**U**	**M**
Accurate SAM Model	***out_1***	36	15	38	12	39	9
	out_2	8	191	6	194	6	196
Sabotaged SAM Model	***out_1***	57	26	60	21	63	17
	out_2	10	157	8	161	6	164

Table 5.9: The Percentage (%) Rates of True Positives Rates (TPR) and False Positives Rates (FPR) of the Planning System under Probabilistic Conditions Using Accurate and Sabotaged Navigation SAM Models.

	Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
	TPR	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
Accurate SAM Model	95.98	29.41	97.00	24.00	97.03	18.75
Sabotaged SAM Model	94.01	31.33	95.27	25.93	96.47	21.25

Table 5.10: Accuracy and Precision of the Planning System under Probabilistic Conditions Using Sabotaged and Accurate SAM Models in Navigation Actions. The Results are Given as Percentage (%).

	Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
	Accuracy	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Precision**
Accurate SAM Model	90.80	92.72	92.80	94.17	94.00	95.61
Sabotaged SAM Model	82.80	84.41	86.00	86.56	88.40	88.65

5.5 Discussion

The results reported in this chapter show that the semantic knowledge domain can effectively help robots to achieve good performance in generating symbolic plans for robot tasks in uncertain situations. When the planning considers uncertainty in world states and action effects, the performance is even better than that of a deterministic plan (Chapter 4). This demand is validated by the high rates of true positives (TP) and the low rates of false positives (FP) achieved in task planning for two different types of actions. The improved performance resulted from the fact that the expectations are not Boolean in manner, i.e. true, false or ambiguous.

Furthermore, in probabilistic planning, decisions about whether the plan of actions has failed or succeeded are based on how likely it is that each expectation will be verified or violated. These decisions are based on the given evidence information.

The drawback of probabilistic planning is that it is unable to detect failure cases when the expected object has a similar description to the planning time object. A typical example of such cases in the experiment is when the robot planner is expected to generate a *take* action to take a glass of water but ends up generating an action to take a bowl instead. Such situations will always result in an 'ambiguous' planning result. To recover from such an issue, concepts should be defined to be totally exclusive, by providing more restrictions involving the properties of their instance objects.

Similar situations arise when both the expected object and the planning time object are instances of the same class. An example for this could be if the robot asked to bring book1, but ends up in bringing book2. This is typically a situation that cannot be handled by considering only general semantic knowledge. Extra information should be provided about the expected object (e.g. the size of book1), or properties related to it (e.g. book1 is a green book) to give it the ability to deal with such unexpected situations.

It should be noted that the performance of probabilistic planning is also reduced when applied in such situations, but to a lesser extent. The resulting probability of action outcomes is highly influenced by the prior probabilities given in the action model. In the case where the expected and the planning time objects are instances of the same class, the last probabilities of the outcomes will always be the same as the prior ones.

5.6 Summary

In summary, semantic knowledge is used to compute implicit expectations for each possible action outcome. Then, a probability distribution is estimated by asking the trained MLN model to investigate how the world state appears while taking into account the implicit expectations and their probabilities. Moreover, a probabilistic MLN model is provided to reason about uncertainty in a given world state. Thus, the result of planning is a probability distribution over the different action outcomes.

The use of an MLN to model uncertainty in acting gives a well-founded treatment, but providing the needed probability values can be a difficult task for large domains. In this implementation, a Markov approach is taken to interpret probability values as measures of belief. The task of providing conditional probabilities for the SAM model is simplified by making assumptions that allow it to use well known learning algorithms to train the MLN. This enables it to encode the probability of classifying the objects as well as misclassifying them when they are seen.

Chapter 6

Semantic Task Planning Based on Information Gathering

6.1 Introduction

Chapter 4 presented the planning process, which depended on the use of an immediately available semantic knowledge base and evidence information to evaluate and check implicit expectations regarding the effects of the generated actions. The outcome of the evaluation enables the assertion of whether these expectations are confirmed, violated, or ambiguous. When the result of the evaluation is ambiguous, this means that the planning system cannot determine whether expectations of the object are matched with the planning time object or not. This may be due to the knowledge and evidence bases containing only a few properties and relation details about the locations or objects.

This chapter will tackle the problem of mismatch between implicit expectations and evidence. This chapter extends the deterministic planning system developed in Chapter 4 to deal with conditions suffering from incomplete information by using the information gathering approach.

An effective information gathering scheme for modelling and reasoning about uncertainty due to lack of information is presented. This scheme is used to collect the necessary information to support the assessment of the implicit expectations with ambiguous truth values. The treatment in this case does not include probabilities. While a probability is used to choose the information that is likely to achieve the highest reduction in uncertainty and then plan to collect it.

This chapter will use the planning system to generate the necessary actions that the robot can use in order to reduce or eliminate uncertainty due to incomplete information. This choice is motivated by the fact that the planning system can be provided with the ability to handle complex situations involving a lack of information in an effective and automatic way. The concept stems from the ability to model the occurring situation as a planning task according to the information available. In this regard, the initial state of the task planning is represents a situation of incomplete information, while the goal represents a situation where that information is available. Therefore, the generated plan includes the necessary movement, exploration, searching or matching actions needed to gather the required information.

6.2 Motivations

In this section three key concepts are defined which concern information gathering for task planning:

1. The planner's initial set of information about the world is incomplete. When the size and nature of the robot domain is considered, it is not possible to assume that the planner will have gathered all the information needed to devise a plan.
2. The robotic system should gather the necessary information through planning. While all the information relevant to a problem may not have already been gathered, it is desirable for the planning system to have the ability to gather new information as needed. The relevance of possible information should be determined during plan generation, so it makes sense to gather that information through planning.

3. The planning system may not return the required information quickly, or at all. Executing plans to obtain the information will typically take more time than the planner would spend generating plans. In some cases, it will not be known a priori which action gives the necessary information, and it will be necessary to search a planning domain to find the appropriate action. However, it may not possible to find those actions at all. In this case the system should not stop planning while waiting for answers to its queries, but should keep planning and looking for other possible plans that do not depend on answering those specific queries.

As a motivation example, suppose that the robot is in the bedroom *r5* and it has been ordered by the user to clean the dining room *r4* before dinner time. To achieve the task, the robot task planner generates a plan that could include the following actions:

(*move(robot,r5,hall) ;move (robot,hall,r4) ; clean(robot, r4)*).

Now suppose that the robot planner has just finished the generation of the *move(robot, hall, r4)* action, and the checking process is triggered to see if the specifications of *r4* are matched to its specifications in the knowledge base. Assuming that the problem domain contains only the available properties of *r4*, then the knowledge base is queried about the implicit information of being in the dining room. The knowledge base will depend on its available information to answer the queries.

Suppose that all the robot had stored in its entire knowledge database was a chair and a table. As chairs and tables can be found in rooms of different types, this information is not enough to help the planning process to establish whether *r4* is a dining room. Therefore, the initial result of the checking process is unmatched. At this stage, this

process can assume that the robot is in room *r4*, since it could not find any evidence against *r4* being a dining room.

If the robot is to be sure that the task is planned successfully (i.e. the robot does not end up cleaning a room that is not the desired one), it needs to gather more information to help it establish that it is in the right room. It is important to look for information which is needed to decide whether the implicit expectations of being in the dining room are verified or violated. For example, the planning system might generate a plan to gather information which specifies with low probability of finding a sink in a dining room, in order not to violate the dining room definitions. It can also look for a dining table, as a dining room is expected to contain at least one dining table.

Considering that the amount of available information about the robot environment is large, a planner should gather the information as needed by the planning process. Since gathering information may take a considerable amount of time, it would be wise for the planner to continue planning while the queries are being processed.

6.3 Gathering Information by Planning

The issue faced by the robot in the above example was an unexpected situation that arose due to lack of information. The information is needed to determine whether the robot planner has scheduled its actions successfully. It cannot always be determined whether some implicit expectations hold true or not.

In the following, a careful approach will be taken whereby the planning system must search for information required to support the planner to generate its plan and assess

such expectations and to establish whether the actions have been successfully scheduled. The information gathering approach which is proposed in this chapter is based on automatically analysing and representing an unexpected situation as a planning problem with incomplete information about the robot world state.

Planning is used to solve the problem at hand. The solution is a plan that includes a schedule of actions designed to collect information for supporting task planning, and to determine whether implicit expectations are met or not. The procedure of handling a state involving lack of information using planning can be summarised as follows:

1. **Case estimation**: when a state of lacking information appears, it is analysed with the aim of creating a planning problem from the specification of an initial state and a goal to achieve. In this stage, the initial state of incomplete information problem encodes, as a set of assumptions, the properties of objects and places that currently have ambiguous truth values. These assumptions represent one possible assignment of truth values to objects and places properties. Conditions under which the planning system needs information must also consider the relationships between objects and places.

2. **Plan generation**: the created planning problem is then passed to the planner so that it can be solved by generating a plan containing, for example, movement, exploration and any other information gathering actions. The movement actions are intended to put the robot in a state where it can execute the information gathering actions. For instance, this could mean moving to a location where it is possible to observe other parts of a room. An example of an explore action is the rotating of an object held by the gripper of the robot in order to record a hidden side of that object.

3. **Updating**: updating of the planning domain or the problem based on the new information which is obtained from step 2.

4. **Querying**: the action of querying the knowledge base to gather information during planning about the state of the world before the planning began (i.e. about the initial state).

6.4 Predicting Information Gathering under Uncertain Planning

This section deals with planning under uncertainty in predicting, i.e. the prediction if IG can solve the problem of lack of information in acceptable criteria or not. The primary concern here is the need for feedback from the planner whilst its works to generate plans. During planning time, a world state may encounter a lack of information, so the planner cannot continue with finishing the plan. In this case the decision maker is prepared to deal with any state it may encounter (O'Kane et al. 2006).

6.4.1 Decision Making

Consider the problem of making a decision when there is a lack of information regarding the world state. This case is modelled as a decision to be made by the decision making unit. To formalise this decision problem, the following entities should be defined:

1. A set of robot actions, $\boldsymbol{A}$, that have the ability to add information into robot world states (information gathering actions).

2. A set of decision maker parameters, $\boldsymbol{\Theta}$. This set should encode all of the lack of information (or uncertainty in the case of probabilistic conditions) in the world state input to the decision maker.
3. A cost function, $\boldsymbol{L}: \boldsymbol{A} \times \boldsymbol{\Theta} \rightarrow \mathbb{R}$, which encodes the relative undesirability of each possible case. This is the quality that should be minimised. Inversely, it can be useful to find a reward function that should be maximised.
4. A lack of information (uncertainty) model for $\boldsymbol{\Theta}$. This is the possibility of the needed $\boldsymbol{\theta}$ (or distribution $\mathbf{P}(\boldsymbol{\theta})$ under probabilistic uncertainty).

The objective function is to choose an action, a, that will result in the smallest possible $L(a, \theta)$ and provide the necessary information in the robot world state. Under lack of information (or uncertainty), the best course of action is to consider the worst case cost. The worst case optimal decision, a^*, is given by:

$$a^* = \underset{a \in A}{argmin} \ \underset{\theta \in \Theta}{max} \ L(a, \theta) \tag{6.1}$$

With the probabilistic uncertainty model, the choice of θ is random, so the relevant measure is the expected cost. The decision 'a^*' that minimises the expected cost is:

$$a^* = \underset{a \in A}{argmin} \ E_\theta[L(a, \theta)] = \underset{a \in A}{argmin} \ \sum_{\theta \in \Theta} P(\theta) \, L(a, \theta) \tag{6.2}$$

In either case, a plan is simply a choice of some $a \in A$.

6.4.2 Including an Evidence Space

The previous formulation did not give the decision maker any special information about what selection would be made for θ on a particular task. This model is extended by including an evidence space V. Each $v \in V$ corresponds to a measurement or reading that is given to the decision maker to facilitate the selection of θ. In case of probabilistic planning, this depends on the properties or probabilities of objects and places returned from the trained MLN in the inference phase.

The decision maker is given some value of $v \in V$ and can use this value when selecting a value for $a \in A$. Thus, a plan is a decision rule (or strategy/policy), i.e. $\gamma: V \rightarrow A$. The presence of evidence changes the models of lacking information (uncertainty) to be conditioned on the value of v. Two cases are using evidence space:

- Nondeterministic: assume that $\boldsymbol{v}$ restricts the set of choices available for θ. This can be expressed as a function $\boldsymbol{F}: \boldsymbol{V} \rightarrow \mathbf{2}^{\boldsymbol{\Theta}}$ so that $\boldsymbol{F}(\boldsymbol{v}) \subseteq \boldsymbol{\Theta}$ represents possible choices for θ given $\boldsymbol{v}$. Now the optimal decision rule $\boldsymbol{\gamma}^*$ is simply the one that makes the best worst-case decision for each $\boldsymbol{v}$:

$$\gamma^*(v) = \underset{a \in A}{argmin} \; \underset{\theta \in F(v)}{max} \; L(a, \theta) \tag{6.3}$$

Notice that the only change from (6.1) is that the max operation is over only $F(v)$, i.e. v is related to θ, rather than all of Θ.

- Probabilistic: The distribution for θ is now conditioned on $\boldsymbol{v}$. That is, for each $\boldsymbol{v} \in \boldsymbol{V}$ and $\theta \in \Theta$, the conditional probability $P(\theta | \boldsymbol{v})$ is known. Given $\boldsymbol{v}$ and $\boldsymbol{a}$,

the expected cost (also called conditional Bayes risk in this context) is expressed as:

$$\mathrm{E}^{\theta}[L(a,\theta)] = \sum_{\theta \in \Theta} \mathrm{P}(\theta|v) L(a,\theta) \tag{6.4}$$

The decision rule (comparable to a cost function in machine learning) to minimise this is given by:

$$\gamma^{*}(v) = \underset{a \in A}{\operatorname{argmin}} \sum_{\theta \in \Theta} \mathrm{P}(\theta|v) L(a,\theta) \tag{6.5}$$

Two common examples of decision-making with evidence are parameter estimation (Berger 1985; Degroot 2004) and classification (Devyver and Kittler 1982; Duda et al. 2000; Mitchell 1997). In both cases, they have $A = \Theta$ and $L(a, \theta) = 0$ if and only if $a = \theta$. The observation v will give some information about θ, perhaps as a feature vector of θ.

6.5 Gathering Information from Knowledge Bases

The definitions for variables, constants (i.e., objects), logical atoms, states, actions, operators, and plans are presented in a STRIPS style. A world state is complete if it contains all of the atoms that are true in the environment; otherwise, it is incomplete. A plan is a sequence of actions (i.e. ground operator instances). In STRIPS, a plan is denoted as an ordered set of a ground instances of an operator, for example $\{o_1,$

o_2,...,o_k}, where each o_i is a ground instance of an operator and every ground operator is an action a_i in STRIPS style. Thus, Plan (Π) = {a_1, a_2, ..., a_k}.

When the planning system requires more information about the task at hand, it issues a query. In this chapter, the expression of the query is of the form:

$$Q(x') \leftarrow C(y') \tag{6.6}$$

where Q is the unique label of the query, C is a conjunction of (possibly ungrounded) literals, y' is the set of variables appearing in C and x' is the set of variables whose values are being searched. In this study, it is required that $x' = y'$. The Q is said to be grounded when $x' = y' = \emptyset$, i.e. no variables in the query expression. The intent of a query is to gather information during planning about the state of the world before planning started (i.e. about the initial state).

For example: suppose the robot is ordered to find within the house environment the kitchen, where an oven is located in order to cook the food. Suppose also that the robot knowledge base has types such as *Oven*, *Kitchen*, and *Cooking*, along with the relations *isContained*(*Kitchen*,*Oven*), *isLocatedIn*(*Oven*, *Kitchen*), and *isProvided*(*Oven*, *Cooking*). Given these relations, the corresponding query can be written as:

Q(*O*,*K*) ← *Oven*(*O*) ∧ *isLocatedIn*(*O*,*K*) ∧ *Kitchen*(*K*) ∧ *isContained*(*K*,*oven1*) ∧ *isProvided* (*oven1*,*cooking*)

where *oven1* and *cooking* are objects, and *O* and *K* are variables. The answer to the query $Q(x') \leftarrow C(y')$ in a state *S* is a set of variable substitutions = { ρ_1, ρ_2,..., ρ_n },

where ρ_k specifies a value for each variable in x such that all the positive atoms in $\rho_k(C)$ are in state S and none of the negative atoms in $\rho_k(C)$ are in S.

A knowledge base (KB) is defined to be any external source, such as the SKB or the evidence database that can provide information during planning. The specification of a knowledge base describes the contents of the information provided. More formally, the specification of a knowledge base is expressed as:

$$\mathrm{KB}(x^i, x^o) \rightarrow B(y) \tag{6.7}$$

where KB is the unique label of the knowledge base representing the head of the formula, B is a conjunction of logical atoms representing the body of the provider description, y is the set of variables appearing in B, x^i is a set of input variables, and x^o is a set of output variables. If $y = x^i \cup x^o$, then the knowledge base formula is completely explicit. In this chapter, it is required that the knowledge base specifications should be completely explicit.

Example (Knowledge base). Suppose that there is a knowledge base that returns all the rooms (R) located in a house (H) for any given house. The specification of this provider would be: $\mathrm{KB}(H,R) \rightarrow House(H) \wedge hasRoom(H,R) \wedge Room(R)$.

The explanation for the requirement that KB must be completely explicit is that the answer to a query is a value assigned to a variable. This value is defined for each variable in the head of the query. That variable substitution is applied to the body of the provider description and the ground predicates can be inserted into the state of the world.

A knowledge base (KB) can be queried by supplying a variable substitution V that specifies a value for each of the input variables in x^i. The result of the execution of the KB is a set of variable substitutions, i.e. Result(KB, V) = $\{\rho_1, \rho_2, ..., \rho_n\}$, where each ρ_k is a variable substitution that specifies a value for every variable in x^o such that $\rho_k(V(B))$ is true in the world. This means that the information returned is robust, but may not be complete, i.e. Result(KB, V) may not contain all of the possible substitutions that make $V(B)$ true.

Let X be a set of knowledge bases (in this thesis the knowledge bases are the SKB and the evidence).The total amount of knowledge that can be gathered from these providers can be defined as $\delta(X)$. Formally,

$$\delta(X) = \bigcup_{\rho \in \text{Result}(\text{KB},V)}^{\text{KB} \in X} \rho(V(B)) \tag{6.8}$$

for every variable substitution V that specifies a value for every input variable in KB.

6.5.1 Complete vs Incomplete Information Planning Domain

A complete information planning problem is a tuple $P^C = \langle S^C, G, D \rangle$, where S^C is a complete initial state, G is a goal (a goal state that represents a world state that suffers from incomplete information), and D is a planning domain description that consists of a set of planning operators O. A solution for the planning problem P^C is a sequence of actions (i.e. ground operator instances) that, when executed in the initial state, accomplishes the goal G.

An incomplete information planning problem is a tuple $P^I = \langle S^I, X, G, D \rangle$, where S^I is a set of ground atoms that are initially known, X is a set of knowledge bases, G is a goal, and D is a planning domain description. The total knowledge available about the initial state is given by $S^I \cup \delta(X)$, where S^I denotes an incomplete state of the world, and X is the set of available knowledge bases.

If $S^I \cup \delta(X) \subseteq S^C$, then the incomplete information planning problem P^I is consistent with a complete information planning problem P^C. The results returned from knowledge bases will be assumed to be fixed during the course of planning. Therefore, for a knowledge base KB, Result(KB,V) is fixed and does not change during planning, so $\delta(X)$ is also fixed. This assumption specifies a class of robot task planning problems in which the information collected from the semantic knowledge and evidence is static during the planning process.

Knowledge bases, i.e. knowledge bases (such as the SKB and the evidence) are defined as a database that returns information about the initial state only, and does not have any world-altering effects. In this chapter, a new type of SAM is developed, is called Semantic Action Model for Information Gathering (SAM_IG) , which has the ability to return knowledge from KB provided that the inputs, outputs, preconditions and effects of the action are known. The output parts of the new SAMs_IG are related to enrich the knowledge of the incomplete information problem.

Example 3 (Knowledge providing by SAM_IG action).

By referring to example 2, the semantic description of SAM_IG is as follows:

define atomic process FindRooms(
inputs: (h-house),

outputs: (r-room),
precondition: ( ),
result: (hasRoom(h, r))

The information gathering in the semantic knowledge phase is formalised as a tuple $P^S = \langle S^I,\ W,\ C,\ K \rangle$, where S^I is an (possibly incomplete) initial state, W is a set of knowledge providing models (KB) that are available during the planning process, C is a goal, and *K* is a collection of SAMs_IG' process models such that $C \in K$. A solution for P^S is a sequence of atomic SAM_IG processes that, when executed, achieves the functionality desired by the process C.

P^S is equivalent to an incomplete information problem $P^I = \langle S^I,\ X,\ G,\ D \rangle$, where *G* is the goal C, D is the planning domain description generated by using the translation algorithm of Chapter 4 (i.e. by applying Algorithm 4.1 on *K* to produce PDDL), and *X* is the knowledge base specifications for actions in *W* generated as explained above (i.e. $\delta(X) = \delta(W)$). The Information Gathering (IG) algorithm (Algorithm 6.1) uses the knowledge base specifications to decide which action provides relevant information during planning. Figure 6.1 shows the mapping process between the semantic knowledge phase and the process of gathering information from the incomplete-information state space.

Algorithm 6.1 shows the IG algorithm which is used to decide which action (SAM_IG) provides relevant information during planning when information is incomplete. The algorithm starts as soon as incomplete information detected. The incomplete information is the goal that should be verified by world state in order, for the main planning task, to continue its work. So, a temp is generated from the goal (G) and the Plan which returns the necessary sequence of actions that solved the problem.

The G is partitioned into g_i, which represents the sub goals that should be verified. If the status of g_i is verified, then searching for the SAM_IG action that have g_i as a precondition in addition to the other preconditions of the action in order to create query to access the knowledge base to extract the necessary information to enrich the world state with new information. Then a sequence of SAM_IG is returned as a plan to the planner to execute it and recover from incomplete information.

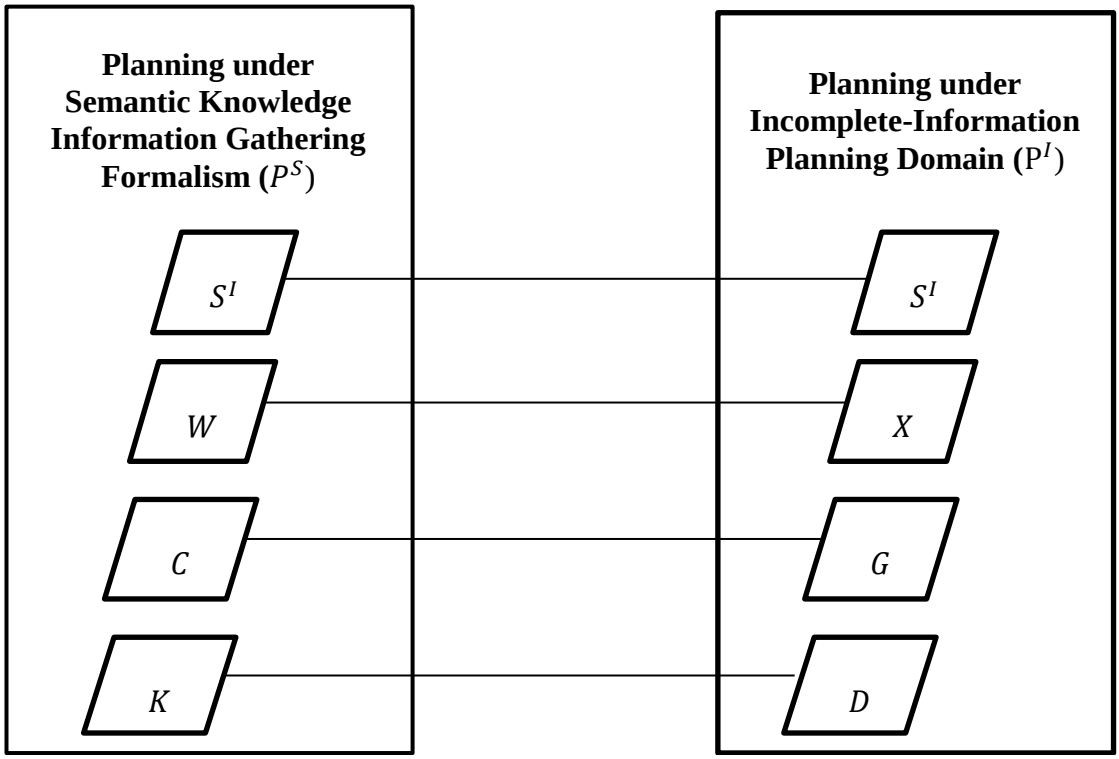


Figure 6.1: Mapping between Semantic Knowledge Information Gathering and the Incomplete Information Planning Domain

```
Algorithm 6.1
IG (P^I) = Plan
Plan = 0
temp = (G,Plan)
Loop
        If temp is Null then return failure
        Take G from temp and remove it
        If G  is Null the return Plan
        Loop from i = 1 to i < |G|
                Choose g_i from G
                        If status(g_i) is true
                                Let (a) be the action which g_i in its preconditions
                                If no such action then return failure
                                Let (pre) be the preconditions of action (a)
                                D(y) = KB(g_i,pre)
                                Append (a) in Plan
                                S^I = S^I+D(y)
Return (Plan)
```

6.5.2 Semantic Action Model for Information Gathering (SAM_IG)

Semantic action models for information gathering are new type of actions which are concerned in the purpose of accessing the knowledge bases and obtaining new information. This information is used to support the robot planning system to recover from unexpected situations which are caused by lack of information.

SAM_IG is applied on the head of the equation (6.7), i.e., KB(x^i, x^o) which consists the input and the output query variables. These variables are related to process part of SAM_IG and from the type of these variables the knowledge bases can be specified. Then the body part of the query ($B(y)$) is grounded from the related knowledge bases.

Figure 6.2 explains the process part of the *find* action for information gathering.

```
<process:ProcessModel rdf:ID="FIND_PROCESS_MODEL">
<service:describes rdf:resource="#FIND_ACTION"/>
<process:hasProcess rdf:resource="#FIND_PROCESS"/>
</process:ProcessModel>

<process:AtomicProcess rdf:ID="FIND_PROCESS">
<process:hasInput  rdf:resource="#_KP"/>
<process:hasInput rdf:resource="#_TARGET"/>
<process:hasInput rdf:resource="#_TOPIC"/>

<process:hasOutput rdf:resource="#_TARGET"/>
</process:AtomicProcess>

<process:Input rdf:ID="_KP">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
file:/project/ontology/world.owl#KP</process:parameterType>
<rdfs:label></rdfs:label>
</process:Input>

<process:Input  rdf:ID="_TOPIC">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
file:/project/ontology/world.owl#Place</process:parameterType>
<rdfs:label></rdfs:label>
</process:Input>

<process:Input  rdf:ID="_TARGET">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
file:/project/ontology/world.owl#Room</process:parameterType>

<process:Effect  rdf:ID="_TOPIC">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
file:/project/ontology/world.owl#Place</process:parameterType>
<process:has rdf:resource="#_TARGET"/>
<rdfs:label></rdfs:label>
</process:Effect>
```

Figure 6.2: Semantic Action Model for Information Gathering SAM_IG.

6.6 Planning Process

STRIPS-style planners (Metric-FF, LPG-td) require the specification of an initial (belief) state and a goal formula as input. The following subsections will discuss these requirements and how they are represented in information gathering situations.

6.6.1 Initial Belief State

Generating plans to successfully collect information implies that (i) the planner takes into account the issue of lack of information in the robot environment and (ii) the initial

state of the planning domain is incomplete. To this end, belief states are used to represent the robot's incomplete and uncertain knowledge about its world at some point in time, i.e. a belief state represents a set of hypotheses about the actual state of the world given past evidence information, i.e., hypothesis (expectation|evidence).

The initial belief state contains hypotheses about the truth value of each expectation that must be checked. This is accomplished by asserting that the expectations can be true or false. The initial belief state also includes other information needed for the planning task, e.g. the robot's whereabouts, knowledge about the workspace, etc.

The new gathered information is used in two ways: firstly to enrich the initial state with new information to support the planner to generate plans, and secondly to evaluate the truth values of the implicit information which is related to the effects of the plan actions.

Example Suppose that the robot planner has just decided to insert the action *move(robot,hall,r1)*, where *r1* is an instance of *BedRoom*, which is defined as follows:

Bedroom is a *Room* and *isContained* (min 1 *bed* and exactly 1 *sofa*))

Suppose that the planner could establish (by depending on the problem definition) that the final location of the robot will be a room, with its explicit expectation room *r1* verified. Suppose also that the KB has not recorded any sofa or bed inside this room, so that the implicit expectations corresponding to the constraints (*isContained* (min 1 *bed* and exactly 1 *sofa*)) are not known to be held or violated.

The belief state has four hypotheses (possible worlds) that result from the different combinations of the truth values of the expectations. Notice that the symbol *r1* in this

situation is a temporary symbol that refers to the planning time location (or object) of the robot, which might be different from the expected location.

The process in charge of creating the initial belief state needs to incorporate other information that is needed to create the task plan. This includes a symbolic description, within the knowledge base, of the places , that the planning system is likely to find individual objects or features related to an ambiguous implicit expectation. For instance, the planning system can generate a plan to access the knowledge bases and hence provide new information to the problem initial state to support the planner in generating the task plan. This plan also can find any object related to its room or any negative evidence object. Thus, *isContained*(*bedroom*, *bed*) etc., will be added to robot initial (belief) state.

Another solution will depend on the information sources to be queried, i.e.:

$$\mathrm{KB}(bed, room) \rightarrow isContained(room, bed).$$

This applies to all rooms which have beds in them. The planning system can then decide if *r1* is one of the bedrooms or not.

6.6.2 Goal Specification

The goal state to be reached by the plan is specified by a modal formula containing a conjunct of the predicates associated with the expectations whose truth values are ambiguous. The goal represents the body part ($B(y)$) of Equation (6.7). The predicates are assigned the expected truth value of their corresponding expectations. For the

previous example, the goal formula that could be passed to the problem domain of the planner is:

isContained(*r1*,*bed*) and *isContained*(*r1*,*sofa*)

which expresses that the information gathering plan should try to reach a state where room *r1* has a bed and a sofa.

6.6.3 Plan Generation

Both planners (Metric-FF, LPG-td) were used to generate the information gathering plans. They take as input an initial belief state and a goal formula, and they generate conditional plans can be applied to the initial belief state to result in a state where the goal formula is verified.

To be able to resume the task at hand within the scope of the top-level task plan, information gathering plans are restricted to include only actions that do not alter the state of the top-level task plan in any relevant way. For example, the information gathering plan to verify the implicit expectations of room *r1* is not allowed to include actions to search for information about other rooms. This restriction is sufficient in navigation scenarios that involve only observation and movement actions, where the top-level actions are used only to move to a certain room.

To prevent the information gathering process from altering the robot world state, one flexible approach is to restrict the process to hold certain conditions at the end of the information gathering process. For example, in the manipulation scenario, when the goal is to take a spoon from the kitchen, the world state should contain a predicate that

requires the robot to be in the same room. This would make it possible to include actions to explore other rooms in the KB, to acquire the possible new location of the spoon outside kitchen.

Example The following plan is generated by the LPG-td planner for checking whether *r1* is a bedroom. It begins with the situation where the truth values of the implicit expectations (*isContained*(*r1*,*bed*) and *isContained*(*r1*, *sofa*)) are ambiguous.

findObjectIn(*bed*,*r1*), *findObjectIn*(*sofa*,*r1*)

This results in a search of the KB to verify the truth value of the restrictions (*isContained*(*r1*,*bed*) and *isContained*(*r*,*sofa*)) and to change the value of the ambiguous implicit expectations to matched or unmatched.

Note that the fact that the plan includes movement and searching actions is specific to navigation scenarios and is not a restriction. In other scenarios movement actions might not be needed, but searching actions will always be necessary, since the aim is to gather information. For example, if the planner generates the action *grasp(robot,c21)*, where the symbol *c21* refers to a cup that contains coffee, the searching plan would include actions to check the contents of the cup, but no actions to change the location of the cup.

The planning system concludes that the implicit expectations are verified when the last planned action of the information gathering plan is successfully completed. Reaching a failed action implies that there was at least one violated expectation.

6.7 Information Gathering for Planning under Probabilistic Cases

Chapter 5 discussed situations where the evidence is not sufficient to compute a probability of the possible world that matches what was predicted by the task planner. For instance, the planner might have predicted that, after the construction of a *move* action, the robot will be either in the living room *r4* or in the kitchen *r5*. Thus, if the robot evidence database does not have anything to support its decision, then the planning process will not be able to determine which room the robot is in with any precision.

The probabilistic planning process (which was discussed in Section 5.3) can select the outcome with the highest possible world probability as the resulting outcome. Chapter 5 also mentioned that the outcome with the highest posterior probability that exceeds a given restriction, i.e. (Std_{dev}+ mean), can also be selected. However, there can be cases where no outcome satisfies the criterion due to the high uncertainty in the computed possible world related to the action outcomes.

As in the case of deterministic planning, the robot can gather useful information in order to reach a situation where uncertainty in the action outcomes or world states is minimised or even removed entirely. This section presents an alternative approach which explains how such information can be specified without planning all the way to a state where all the uncertainty is eliminated.

The principle of the approach is to choose the information that is likely to achieve the highest reduction in uncertainty and then plan to collect it. In Section 6.8.2, a test scenario is presented where information gathering is applied to reduce the uncertainty surrounding the possible world of a navigation action.

6.7.1 Information Gain

One measure that can be used to select the information to gather is called information gain. This measure expresses the average reduction in the uncertainty of a random variable. More formally, the gain in information achieved for the random variable *R* (which denotes the possible action outcomes,) with known values of an observation (evidence) variable O_i is given as follows:

$$\mathrm{GI}\,(R|\mathrm{O_i}) \;=\; \mathrm{E}\,(R) - \mathrm{E}\,(R|\mathrm{O_i})\,, \tag{6.9}$$

where the quantity E(*R*) is known as the information entropy. This is the average amount of information contained in a random variable. E(*R*|O_i) is the conditional entropy of *R* given O_i. The quantity E(*R*) is used to measure uncertainty in a discrete random variable *R*, so it is hence defined as:

$$E(R) = -\sum_{r\in R} p(r)\log(p(r)). \tag{6.10}$$

The conditional entropy E(*R*|O_i) represents the average uncertainty in *R*, taking into account that the value of O_i is known, i.e.:

$$E(R|\mathrm{O_i}) = \sum_{\mathrm{o_i}\in\mathrm{O_i}} p(\mathrm{o_i})E(R|\mathrm{O_i} = \mathrm{o_i}) \tag{6.11}$$

The conept of information entropy was introduced by Shannon (2001). Information entropy is as a measure of the information contained in a message composed of a finite set of symbols.

For the purpose of information gathering to reduce uncertainty about action outcomes and world states, all observation variables should be taken into account to computing the information gain. In other words, Equation (6.9) is applied to all observed random variables O_i, and the variable that gives the highest information gain is selected as the useful information to collect. If *obs* is the observation to be collected, then *obs* is determined as follows:

$$obs = \arg\max_{O_i} GI(R|O_i) \tag{6.12}$$

Once *obs* = O_k is determined, the planner can generate a plan to look for objects that are instances of the class C_k. For instance, if Equation (6.12) reveals that *obs* = O_1, which denotes the sink object, then the planning system can issue a query to ask the knowledge base about all the rooms that contain a sink.

A decision-theoretic approach that takes into account the cost of information gathering, and can be selected by defining an utility function *U*. For instance, (Takeuchi and Furuhashi 1998) define the utility associated with a sensing action as a decrease in uncertainty about the robot location, where uncertainty is measured by entropy. The cost of each sensing action is then subtracted from its utility, and the action that has the highest expected utility is thereby selected.

In this case, it is required to associate an utility with each observed random variable O_i. This can be measured by the information gain function specified in Equation (6.9). As a result, the observation that achieves the highest difference between its utility and its cost is selected as the information to gather. Given the utility $U(O_i)$ and costs $C(O_i)$ of all observations O_i, the robot hence selects the observation *obs* as follows:

$$obs = \arg\max_{O_i}(U(\mathrm{O_i}) - C(\mathrm{O_i})) \tag{6.13}$$

where $U(\mathrm{O_i}) = \mathrm{GI}(R|O_i)$. Once the information to collect has been identified, a plan can be generated to collect it, for example by searching different rooms in a KB to discover the facts that were initially hidden. The advantage of planning to collect only one piece of information at a time is that the planning problem becomes much less complex.

6.8 Experimental evaluation

The experiments will investigate how well the information gathering algorithm would perform in planning problems where some of the information is completely unavailable.

6.8.1 Information Gathering for Deterministic Planning

This section describes test scenarios that examine the capacity of the deterministic planning process to reason about situations involving a lack of information. The goal was to test the planning approach proposed in Section 6.6, to compute solutions (plans) for gathering the information needed to check implicit expectations.

Information gathering is a careful approach that can be used to deal with conditions when deterministic planning cannot be certain about the truth values of implicit expectations.

Experiment 1. This experiment examines how the number of solutions (plans) found by the planners (LPG-td and Metric-FF) is affected by the amount of information available during planning. In this experiment, the problems on the robot delivery domain are used. In these problems, a robot is ordered to deliver every object in the house to the correct place. The robot needs to gather information from all the rooms in the house regarding the objects' locations to prepare its strategy for solving the problem. Such information is provided by SAM_IG. The goal is to generate a sequence of actions to solve the problem.

In these experiments, the planner was applied to 50 randomly generated problem instances. For each problem instance, the planner was run several times, each time varying the amount of information available about the initial state by varying the following quantities: $|S^I|$, the number of atoms of S^C initially given; and $|\delta(X)|$, the amount of atoms of S^C that were made available through the input knowledge bases X, where S^C is the set of all possible ground atoms in the domain.

The percentage of times that the planner could determine a plan is measured as a function of the quantity: $\frac{|S^I \cup \delta(X)|}{|S^C|}$.

This quotient represents the portion of atoms regarding the initial state that are available during the planning stage. In this experiment, the value of $\frac{|S^I \cup \delta(X)|}{|S^C|}$ is varied from f = 0 % to f = 100 % in steps of 10 %. This was achieved as follows: (i) A set of atoms was randomly chosen for $|S|$ such that the size of this set is equal to the fraction specified by the particular f value; (ii) For each atom in this set, it was randomly decided whether the atom should go into incomplete initial state, S^I, or whether it should be provided from the SAM_IG in X. Using this setup, the planner performed 50 runs for each value of

$\frac{|S^I \cup \delta(X)|}{|S^C|}$. The results in Figure 6.3 show that the success rate of the planner finding the plan increases as $\frac{|S^I \cup \delta(X)|}{|S^C|}$ increases. The planner was able to solve 100 % of the problem instances even when $\frac{|S^I \cup \delta(X)|}{|S^C|}$ was as low as 70 %.

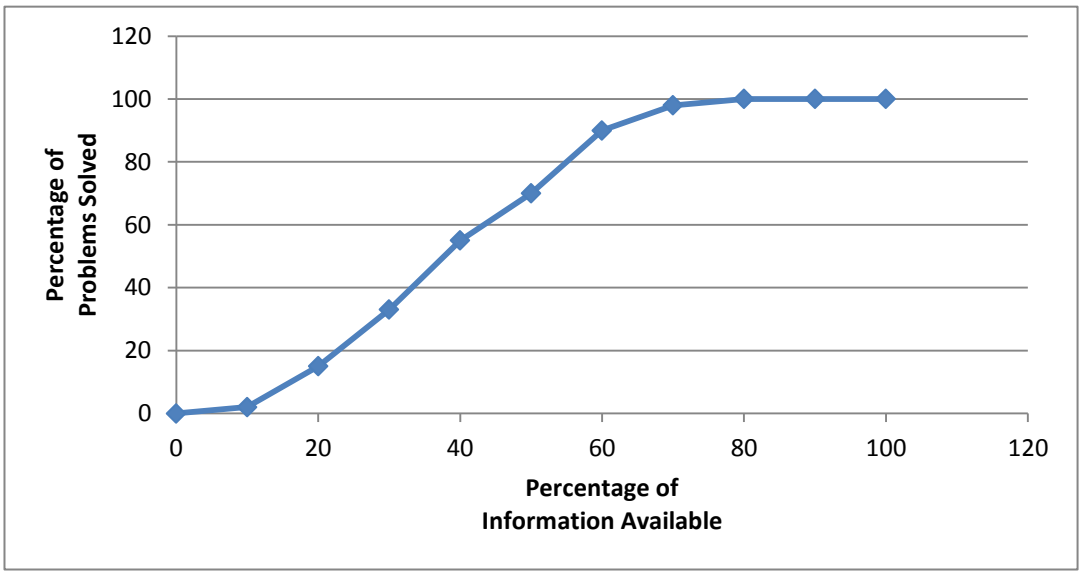


Figure 6.3: The Percentage of Times the Information Gathering Algorithm Supports the Planning System to Find Plans for the Robot Delivery and Arrange Problem as a Function of the Amount of Information Available During Planning.

Experiment 2: In order to recover from the issue of a lack of benchmark systems in generating symbolic plans under the situations suffer from lack of information the evaluation is based on the metrics of false positive rate (FPR) and true positive rate (TPR). Both metrics assume a binary classifier that tries to classify a set of instances as either positive or negative.

The performance of the planning system when supported by the information gathering approach was tested in both navigation and manipulation scenarios. The experiments relating to the manipulation scenario consisted of generating the high-level action *take(robot,obj,place).* This required the robot to pick up the object *obj* that was on the

table by using a robot arm. In this scenario, the information about the types of objects on the table was incomplete, so the planner stopped at that action and asked the IG unit to gather more information about the target object. There are 4 types of object could be on the table, i.e. cup, bowl, milk box and glass of water. The planning system depended on the IG process to enrich the robot work space with new information about the properties of objects on the table in order to verify the implicit information of these objects. The SAM_IG that relate to this scenario is *find* as it is used to query the KB to find more properties related to the target object. Each object was asserted to be one of the objects 20 times, so that the total number of runs was 80.

Similarly, the navigation scenario consists of the action *move(robot,from, to)* to move the robot from a room identified by *from* to another 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 should be specified according to the type of object inside it and the implicit information of being in that room should be verified. Thus, the planning system has been supported by the IG process to extend the initial state of the robot environment with new information. The IG unit used the *explore* SAM_IG in order to query the KB for more information about the target room. Each room type was considered 20 times, so the total number of runs was 160. For each run, the type of the final location of the robot was selected successfully from the 8 available types.

Table 6.1 shows the results according to three different values of (*Th*). The row and column specifications are the same as those explained in Section 4.6.2.2, but this experiment deals with the situation of possessing incomplete information in the world state and the method of using an information gathering process to recover from that situation.

In these results, the planning system with an information gathering process is able to show a high percentage of true positives (positive instances correctly classified) and a low percentage of false positives (negative instances erroneously classified as positive). The true negatives (failure situations) are detected with different percentage for the navigation and manipulation scenarios. High percentages are observed for the navigation task (95 %, 98 %, 98 %), and small percentages for the manipulation task (61 %, 67 %, 74 %).

The favourable results in this case, compared to deterministic planning case (4.6.2.2), comes from the fact that the IG approach extends the robot world state with new information. This information helps the planning system to verify the truth values of the implicit expectations. This leads to minimisation of the number of ambiguous results and maximisation of the number of positive and negative instances.

Table 6.2 presents the true positive rate (TPR) and false positive rate (FPR) for the navigation and manipulation scenarios by taking two different cases for which to handle the ambiguous results. In the first case, the planning system takes the open world approach and considers the ambiguous results as matched because there is no negative evidence in these situations. In the second case, the planning system takes the closed world approach and considers the ambiguous results to be unmatched because it is not specifically determined whether some conditions are held or not.

Table 6.1: Results from Running the Information Gathering Plan Generation for the Actions *find* and *explore*. The Cells Represent Number of Runs that result in Matched (M), Unmatched (U) or Ambiguous (A) Outcomes.

		Th = 3 Objects			*Th* = 5 Objects			*Th* = 7 Objects		
		M	**U**	**A**	**M**	**U**	**A**	**M**	**U**	**A**
Navigation	**M**	27	0	7	31	0	5	32	0	3
	U	0	120	6	0	121	3	0	122	3
Manipulation	**M**	18	0	6	22	0	4	24	0	3
	U	0	34	22	0	36	18	0	39	14

Table 6.2: The Percentage (%) Rates of True Positives (TPR) and False Positives (FPR) for the Information Gathering Plan Generation for the Actions *find* and *explore*. Two World Features are Considered: Open World Treating Ambiguous Cases as Successful, and Closed World Treating Ambiguous Cases as a New Case.

		Th = 3 Objects		*Th* = 5 Objects		*Th* = 7 Objects	
	World Property	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
Navigation	Open	100	4.76	100	2.42	100	2.4
	Closed	79.41	0	86.11	0	91.43	0
Manipulation	Open	100	39.29	100	33.33	100	26.42
	Closed	75	0	84.62	0	88.89	0

6.8.2 Information Gathering for Probabilistic Planning

This section will test scenarios where the probability of the action outcomes computed by the probabilistic planning system involved information gathering. This was discussed in Sections 5.3 and 6.7. A typical scenario where information gathering needed while the planner was generating a conditional task plan for cleaning the living room *r3*, starting from the bedroom room *r1*. This is expressed as follows:

move(*robot,r1,hall*) *; move* (*robot,hall,r3*) *; clean*(*robot,r3*).

The generation of the action *move(robot,hall,r3)* could result in two alternative outcomes. In the first outcome (*out1*), the robot stays unconsciously in hall, with probability 0.1. In the second outcome, (*out2*), the robot moves successfully into room *r3*, with probability 0.9.

As the plan generator requires a clear answer about the location of the robot to continue the generation of a plan, the matching process returned the outcome that had a posterior probability greater than the threshold T = 0.7.

If no outcome satisfied the criterion (the action outcome that used to estimate world state probability was below the threshold T), an information gathering process was launched to gather information that was likely to reduce the uncertainty in the posterior of the outcomes or states. If after the information gathering there was still no possible world (resulted from outcome) that satisfied the selection criterion, the outcome that led to possible world with the highest probability was returned to the plan generator.

When the planner generated the action *move(robot,hall,r3),* it could deal with only one table in the room *r3* as its evidence database contains objects related to the living room. Consequently, the computed possible world of the action outcomes using the inference phase was: P(*out1*) = 0.42; P(*out2*) = 0.58. As none of the outcomes had a possible world probability greater than 0.7, an information gathering episode was initiated to search for objects which would reduce the uncertainty about the location of the robot.

This resulted in a search for objects of type sofa as, on average, observing a sofa was predicted to achieve the highest information gain. Therefore, the planning system issued

a query to the knowledge base through an (IG) algorithm about which of the rooms might contain a sofa.

There are two possible answers from the knowledge base in this scenario. The first is that the knowledge base could return a sofa in *r3*, so that the possible world probabilities of the two outcomes were as follows:

P(*out1*) = 0.18; P(*out2*) = 0.82

As P(out2) >T, the probabilistic planning process returned *r3* to the planner as the resulting outcome of the action *move(robot,hall,r3).* This meant that the next action for the task planner to generate was *clean(robor,r3).*

The second potential knowledge base answer is that no objects were returned during the information gathering phase except for table. Thus, the possible world that was computed before the information gathering would not change, i.e. P(*out1*) = 0.42 and P(*out2*) = 0.58. Because neither outcome had a posterior greater than 0.7, the planning process returned the one that had the highest possible world. In other words, *r3* was returned to the planner as the resulting outcome of generating the action *move(robot,r,r3).*

Figure 6.4 shows the probability of finding a plan in three cases which depend on the *Th* values.

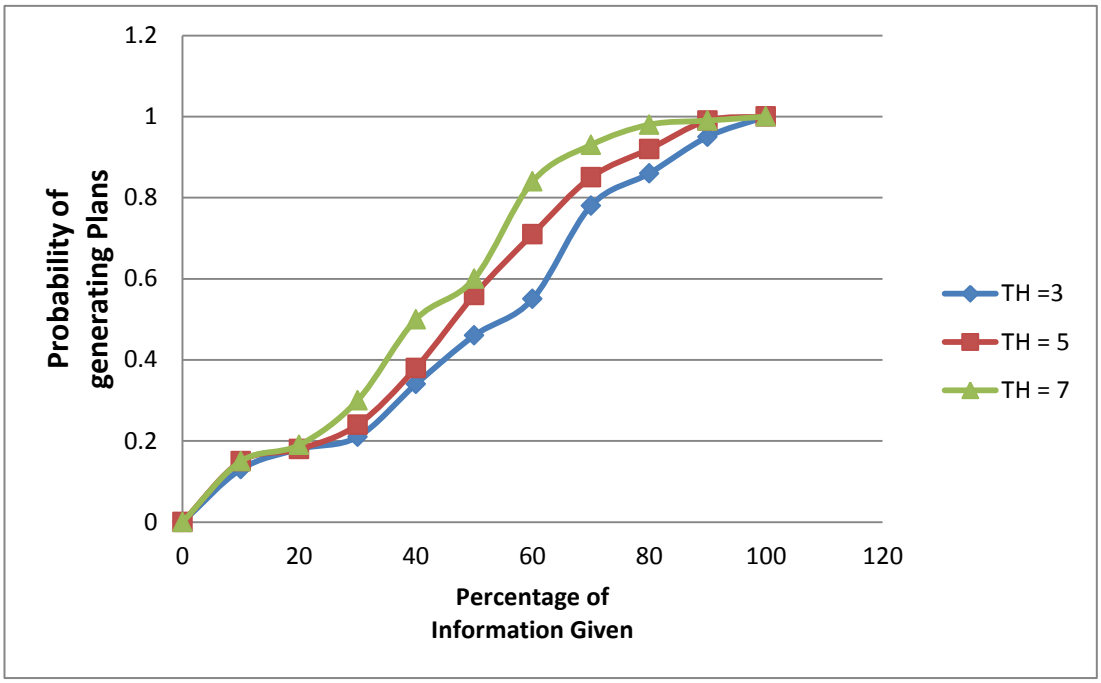


Figure 6.4: The Probability of Finding Plans When the Planning System is Supported by the Information Gathering Algorithm.

6.9 Discussion

This chapter has considered the case when the initial state of the robot domain suffers from a lack of information. A method has been proposed to deal with this problem. The method is based on the concept of gathering new information to enrich the problem space and hence support the robot task planner to continue its work and generate a suitable plan for a given task. The fundamental principle is to model the situation of incomplete information as a planning problem where some information needs to be gathered in order to evaluate expectations with uncertain statuses.

Using the planner for different tasks that involve information gathering has been investigated in several previous studies. For example, information gathering is used to achieve safer robot navigation by (Taniguchi and Sawaragi 2004). However, no work to

date has addressed the use of information gathering based planning to collect information for the purpose of supporting semantic based robot task planning.

The planning system has the flexibility to deal with a variety of unpredictable and complex situations involving incomplete information, which is one of the advantages of using planning in information gathering. The key benefit is that the definition of one planning domain can lead to the automatic computation of solutions for a multitude of situations involving information shortages.

Additionally, when the planner can access semantic domain knowledge, the proposed planning approach can also support the plan execution. This is, according to (Berger 1985), a desirable ability of autonomous robotic systems acting in uncertain environments. In fact, the task planner can reason on a more abstract level (office, kitchen, etc.) to generate the task plan. The execution process depends on that plan, and takes care of other related details at execution-time such as checking and monitoring.

The approach in this chapter differentiates between the information gathering actions (SAMs_IG), i.e. actions that have the ability to access the knowledge bases and extend the robot's initial state with new information from their effect part, from these types of actions that are world-altering (SAMs), i.e. actions that generate effects which change the status of the robot and its environment. The preconditions of information gathering actions include conditions may need to call other actions to validate their truth. Therefore, queries must be issued to the knowledge base to provide information to the initial world or current state.

In this chapter, the actions that provide information will not alter the world because the correctness of the generated plans cannot be guaranteed. This is because the changes

made by the information gathering actions may invalidate some of the steps the planner already takes. However, this restriction is not necessary when the effects of the information gathering actions do not interact with the plan being searched for. In general, safety can be established when the original planning problem and information gathering problem correspond to two disconnected planning tasks that can be accomplished without any interaction. Verifying that there is no relationship between the two problems is a challenging task that will be addressed in future work.

The information gathering algorithm should have the ability to issue queries about any state during planning. This will allow query results to be added to a problem space and allow conditional plans to be generated based on the answers returned by the knowledge base.

6.10 Summary

To summarise the contributions of this chapter, it is important to start from the motivation statement that, it is sufficient to define one planning domain in order to automatically compute solutions for a multitude of cases of incomplete information.

These solutions start from explaining the necessity of using an information gathering approach to recover from situations that suffer from incomplete information. Planning under the uncertainty of incomplete information was discussed and associated techniques for decision making were presented. These included the use of evidence in the case of uncertainty arising from incomplete information.

Modelling the planning problem under complete and incomplete information was explained. A new type of action was modelled which has the ability to access the knowledge bases and obtain new information to extend the robot initial state with new information. This information supports the planning system in dealing with the problem of incomplete information.

The situation of a planning process running under incomplete information has been explained, including the modelling of the initial (belief) state, the goal state, and the plan generation. Information gathering under probabilistic conditions was also explained. This chapter ended with an experimental evaluation that revealed the advantages of using the IG approach to support the robot planning system.

Chapter 7

Planning under Extended Conditions

7.1 Introduction

This chapter describes the approach of generating symbolic plans for robot tasks in situations that suffer from lack of information, in particular, missing information. This approach is different and independent from the approach of information gathering which was explained in chapter 6. Techniques for estimating similar objects to the original object in the plan action are used in the proposed approach in this chapter.

The entire process is characterized by a semantic knowledge realisation, which is performed on robot ontology. In this way it is possible to include semantic information during planning and applied semantic refreshment where it is necessary. After creating new plans, methods are proposed to calculate the quality of these alternative plans in order to select the best suitable one.

The planning system consists of concept relevance operations which are used to detect similarities between objects in the robot world ontology. The concepts of similarity exploit the knowledge which is obtained from ontology to support the planning system in generating useful plans and extend the robot world state with new similar information. This new information with the original information can recover the planner from the condition of lack of information.

In this chapter, the similarity methods are applied on the action details (preconditions and effects) and world states in order to obtain extended action details and world states. There are different similarity measurements that can be used to obtain similar

objects and these measurements are depending on the hierarchical relationships, for instance superclass, and semantic similarity, for instance calculating the distance of the original object from the similar objects. Sections 7.3.2 and 7.3.3 will explain these methods for measuring similarity.

The rest of this chapter is organised as follows: Section 7.2 presents a motivating example; Section 7.3 explains the function of the semantic realisation and refreshment module and the techniques used for measuring the similarity; and Section 7.4 presents the experiments which will be used to validate the approach which is presented in this chapter. Following this, the main outcomes of this chapter are discussed.

7.2 A Motivation

This chapter deals with the same problem that explained in chapter 6, which was the problem of lack of information. This problem means that the robot task planner cannot continue to generate the plan for robot task due to missing information. The approach presented in chapter 6 dealt with this problem by generating a plan which consists of sequence of actions restricted to gather the necessary information from the knowledge bases. This new information was used to check the implicit expectations of the action effects and enrich the robot world states with this new information to support the robot planner to continue its work for generating plans for robot tasks.

The issue with information gathering problem was it took long time to recover the planning system from the situation of incomplete information. The new generated plan, which is directed to solve the problem, delayed the main job of the planner to generate the plan until the information gathering plan is generated and executed. The

information gathering process depends on the planner to generate the plan for solving the lack of information problem.

In this chapter, another approach of solving the problem of lack of information will present. This approach is depending, not on the planner, but on the similarity techniques to extend the robot world states and action details with new similar objects to the original object in the action. The original object in the action was the reason to cause either failure or delay the planner work.

For example, by returning to the same example of chapter 6, i.e., when the user ordered the robot to clean the dining room. So, when the robot started cleaning and its planner generate the action *take*(*robot*,*vacuum1*,*place*), to pick up the vacuum to clean the dining room. The robot faced unexpected situation, where it did not find the vacuum, therefor the robot either to answer fail in performing the cleaning or obtain support from other unit in its planning system.

By depending on the approached that will be explained in this chapter, the robot can find another object which has the similar functionality to vacuum in order to perform its task. This approach depends on the similarity techniques to search in the robot ontology to find which object is the most similar to vacuum, i.e., for example sweeper.

7.3 Recovering from Unsuccessful Planning

When the planning system cannot create a plan for a given task, i.e. when point 4 in Section 4.5.1 reports a failure in generating the plan, a recovery procedure should be used to recover from that situation. This section describes the second technique which is used to deal with unexpected situations, in particular, the situation when the world state is missing some necessary information for the planning system. Chapters 5 and 6

reported the other techniques that were used to recover from unexpected situations. The next subsection explains the role of a new module in the planning system. This module is concerned with realising the semantics of world states and action effects, and refreshing them to endow the task planner with new information to support its work. The following subsection will explain how to extend the initial state with new information. Finally, the last subsection will explain how to test the quality of the alternative plans. Figure 7.1 shows the enhanced planning system architecture which takes into account instances when no plan can be created according to the available world states and planning domain.

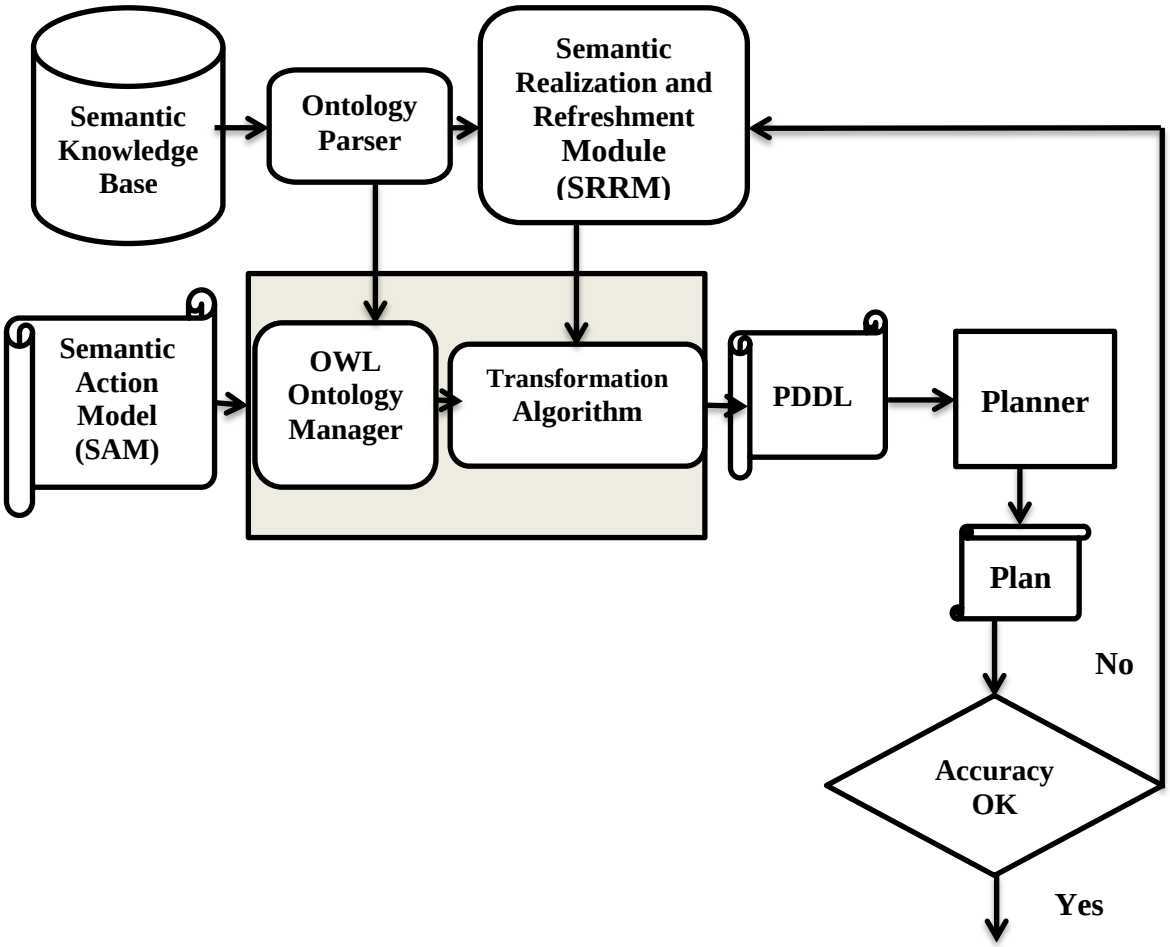


Figure 7.1: Enhanced Planning System Architecture with Semantic Realisation and Refreshment Module (SRRM).

7.3.1 Semantic Realisation and Refreshment Module (SRRM)

Several methods have been developed to analysis semantic knowledge which is stored as ontology. Certain criteria can be used to measure the equivalence or similarity between concepts in the ontology (Hariri et al. 2006). Semantic knowledge, within the details of a SAM, should be invested to recover the planning system from failure situations that occur when generating the plan for a given task. The ontology, which reflects the knowledge base, is used to annotate parameters as inputs, outputs, preconditions and effect components of each SAM. Managing of this ontology requires inference techniques to compute the inferred ontology relationships between ontology concepts. Sometimes it is necessary to get support and feedback from a reasoner such as Pellet DL (Sirin et al. 2007).

Moreover, the criteria for class (actions, objects or places ontologies) relevance are applied to determine semantically equivalent or similar classes to a specific class in the query. The ontology classes are considered relevant if and only if: (i) they have specific hierarchical relationships, and (ii) their semantic distance does not exceed a predefined threshold (Hatzi et al. 2012). The measure of equivalence or similarity can be categorised into two general groups: Lexical Measures and Structural Measures (Hariri et al. 2006). Lexical measures depend on flat similarities such as the Universal Resource Identifier (URI) of entities (the details of this measure is out of the scope of this thesis). Structural Measures consider similarities by studying the kinship of the nodes and structures residing in the ontology graphs.

7.3.2 Hierarchical Relationships

Several possible hierarchical relationships can exist between two ontology concepts. The SRRM supports the following hierarchical relationships between concepts *X* and *Y*:

1. **Equivalent** (*X*, *Y*): The two concepts should have the same URI or they should be equivalent in terms of OWL class equivalence, such as $X = Y$ or $X \equiv Y$.

2. **Subclass** (*X*, *Y*): The concept *X* should be subsumed by the concept *Y*, such that $X \rightarrow Y$.

3. **Superclass** (*X*, *Y*): The concept *X* should subsume the concept *Y*, such that $Y \rightarrow X$.

4. **Sibling** (*X*, *Y*): The two concepts have a common, direct or indirect, superclass *T*, such that $X \rightarrow T$ and $Y \rightarrow T$.

7.3.3 Semantic Similarity

In this work, the SRRM will implement the following two measures to estimate the similarity between concepts in ontology:

1. **Upward Cotopic Similarity**: defines the similarities between concepts according to a set of super classes of the concept in the question including itself. This method can be calculated using the following formula (Hariri et al. 2006):

$$\delta(c_1, c_2) = \frac{|UC(c_1, H) \cap UC(c_2, H)|}{|UC(c_1, H) \cup UC(c_2, H)|} \tag{7.1}$$

where *H* is the ontology hierarchy, and *UC*(*c*, *H*) is the set of super classes of *c*.

2. **Wup Similarity**: measures the similarity between concepts according to the following formula (Wu and Palmer 1994):

$$sim(c_1, c_2) = \frac{2 * d(s)}{d_s(c_1) + d_s(c_2)} \quad (7.2)$$

Where *S* is the least common super concept of c_1 and c_2, $d(c)$ is the (lowest) depth of concept *c* in the ontology, and $d_S(c)$ is the (lowest) depth of concept *c* in the ontology when taking a path through super concept *S* of *C*.

7.3.4 Formal Conditions of Classes' Relevance

Implement a relevancy measure with a certain threshold for the distance between concepts, to define the concepts contained in the ontology with respect to a given concept (query concept).

Suppose that *c* denotes the set of available classes belonging to different ontologies, *H* denotes the set of selected hierarchical relationships, and *T* is the threshold of the distance between classes. In the case of the similarity of depth (Wu and Palmer 1994), *T* is the maximum number of edges present in the shortest path between two classes, with $T \geq 0$. $T = 0$ means there may be no distance between classes, and therefore equivalent mapping is allowed. In the case of the distance upwards cotopic, the threshold *T*, where $T \in [0,1]$, directly reduces the distance between two classes. It defines the maximum distance acceptable between these two classes, i.e. the distance must be $\delta(c_1, c_2) \leq T$.

For each state space (S) of objects, its extended state ES, is defined as the union of all the relevant objects of S (R_o) according to their hierarchical relationships and threshold. This is expressed formally as:

$$ES = \bigcup_{\forall o \in S} R_o \tag{7.3}$$

7.3.5 Extending Initial State and Action Details by Semantic Realisation and Refreshment Module

When the task planner cannot generate a plan for a given task, it is necessary to use certain techniques to recover from this unexpected situation. In this section, a procedure is presented for extending the initial state of the robot world with concepts which are equivalent or similar to original ones. This procedure is used to enrich the world state with new information. The procedure does not stop at this stage, but goes on to extend the action details, which are preconditions, and effect components.

In a pre-planning step, the purpose of the SRRM module is to first realise the semantics of the concepts in the initial states and action details, and then refresh these areas with objects derived from hierarchical relationships and similarity measurements. For the implementation of semantic realisation and refreshment, the produced planning domain and problem are enhanced with semantic information, thus maintaining planner independence in this procedure.

Semantic refreshment is based on the following assumptions:

1. The initial state (IS) is extended with semantically equivalent or similar concepts to the concept in the query; thereby an Extended Initial State (EIS) will be generated.

2. The goal state of the problem remains the same because it depends on initial user request.

3. The action details are enhanced with concepts that are semantically equivalent or similar to the concept in the query of the generated plan, such that the enhancement covers action preconditions and effects. In this manner an Extended Action Set (EAS) is generated.

Algorithm 7.1 summarises all the steps within the Semantic Realisation and Refreshment procedure.

Algorithm 7.1
Procedure: making Semantic Realisation and Refreshment when no exact plan can be generated
Input: I: Initial State,
P={ A_i,…, A_n}: plan,
O={o_1,…, o_k}: set of ontology objects,
T: Threshold
Output: EI: extended initial state,
EA: Extended action details
for i=1 to n
for j = 1 to k
if similar (A_i.hasInput(hasPrecondition),o_j) ≥ T
I = I ∪ o_j
preco(A_i) = preco(A_i) ∪ o_j
end if
if similar (A_i.hasOutput(hasEffect),o_j) ≥ T
effect (A_i) = effect(A_i) ∪ o_j
end if
end for
end for
EI=I;
EA= A_i
End

To better understand the Semantic Realisation and Refreshment procedure, suppose that the initial state *I* and goals *G* of the problem are:

I = {MilkBox, GlassofWater, Kitchen, Fridge}
G = {BottleofWater}

and suppose that there are the following two available operators:

OpenFridge:
prec ={ inFrontOf(Robot,Fridge), at(Robot,Kitchen)}
effect ={OpenedFridge(Fridge)}
TakeGlassofWaterFromFridge:
prec ={ inFrontOf (Robot, Fridge), OpenedFridge(Fridge), at(Robot,kitchen)}
effect ={ Holding(GlassofWater) }

The SRRM for a given combination of hierarchical relationships *H* and a threshold *T* detects the following concepts:

Fridge similar to Refrigerator,
GlassofWater like BottleofWater,

In the pre-planning phase of the planning system, the problem definition is changed as follows:

EIS = { MilkBox, GlassofWater, Kitchen, Fridge, Refrigerator }
EGS = { BottleofWater or GlassofWater }
EAS: OpenFridge:
prec ={ Refrigerator or Fridge, Kitchen }
effect ={ OpenedFridge or OpenedRefrigerator }
TakeGlassofWaterFromFridge:
prec ={ OpenedFridge or OpenedRefrigerator, kitchen }
effect ={ BottleofWater or GlassofWater }

At this point, the planning system will double all operators in size after the extended set of EAS and resulting in the following:

OpenFridge:
prec ={ inFrontOf(Robot,Fridge), at(Robot,Kitchen)}
effect ={ OpenedFridge(Fridge) }
OpenFridge _EX:
prec ={ inFrontOf(Robot,Refrigerator), at(Robot,Kitchen)}
effect ={ OpenedRefrigerator(Refrigerator) }

TakeGlassofWaterFromFridge:
prec ={ inFrontOf (Robot, Fridge), OpenedFridge(Fridge), at(Robot,kitchen)}
effect ={ GlassofWater }
TakeGlassofWaterFromFridge _EX:
prec={inFrontOf(Robot,Refrigerator),OpenedFridge(Refrigerator),
at(Robot,kitchen)}
effect ={BottleofWater }

The new problem <EIS, EAS, EGS> then formalised in PDDL and prompted the planning system to obtain the solution.

7.3.6 Plan Accuracy Assessment

In many cases, the planning system produces more than one plan because of the semantic refreshment and the use of multiple operators in this system. In order to present a good plan, the planning system should be embedded with statistical and quality measures to apply to each plan it generates. Such measures include the number of operations and levels in one plan, as well as quality measures based on distance, which reflect the precision of the plan.

If there is no semantic refreshment during the planning time, then each plan quality is considered approximately equal one. Therefore, these plans are preferable and reasonable as they have fewer levels and the total time of their execution is small.

However, when semantic refreshment is involved, it is important to measure the quality of the plans based on distance of each similar concepts from the original concept. Precise plans are preferred even if they include more actions and levels. To calculate the quality based on a distance, every object that appears in the state space

must be labeled with the semantic distance from the original object and the selected similarity measure.

Additionally, the type of hierarchical relationship that exists between each selected object and the original object is considered, i.e. superclass, subclass or sibling. The Wup measure is calculated as a linear combination of the distances of all the objects that appear in the space.

For a set of N objects, each object class is denoted as c_i, and has an associated semantic distance d_i (obtained from SRRM), and weight r_i (depending on its hierarchical relationship with the original object). The value of the quality measure is hence calculated according to Equation 7.4.

In the case of no exact match between the action preconditions and the world states, the planning system will depend on extended an initial state (EIS) and extended an action set (EAS) to generate alternative plans for the task at hand. This solution works better for cases where the quality of the new plan is high, because the alternative plans have a different quality. It is important for the robot to use the plan with the highest quality.

For the computation of the distance quality metric, each object in the plan is annotated with its semantic distance from the original object and the selected similarity metric. Each object is also annotated with the type of hierarchical relationship it has with the original object (superclass, subclass or sibling). When an exact plan can be generated, the quality of this plan is 1. If an exact plan is not possible, the quality of the plan is calculated using Wup similarity (Wu and Palmer 1994) by:

$$\text{Plan Quality} = \sum_{i=1}^{N} r_i * d_i \tag{7.4}$$

where r_i is hierarchical relationship, d_i is the semantic distance from the original concept, and N is the number of concepts in the plan.

The upwards cotopic distance is calculated as the product of the distances of all the concepts that appear in the foreground, excluding concepts that are equivalent to the original. As before, for a total of N concepts, each concept c_i is marked with a semantic distance d_i, and a weight w_i. The value of the quality measure is then calculated as:

$$\prod_{j=0}^{N} w_j d_i \ , \ d_j \neq 0 \tag{7.5}$$

7.4 Experiments

In this section, two main testing environments are used to validate the methods presented earlier in the chapter. The first environment is based on analysis of the generated plans in main scenarios of planning under deterministic conditions with re-planning.

The second testing environment method is to analyse the performance of the planning system by calculating the planning time under deterministic conditions with re-planning. Then this performance time will be compared with the planning under deterministic, probabilistic and incomplete information conditions. In both testing environments, navigation and manipulation actions are considered.

7.4.1 Environment Setup for Plan analysis

In this environment, the experiments were conducted according to the programming languages and software available to model the robot environment. The SAMs and semantic knowledge base were built using Protégé[4]. The knowledge base is integrated with a Pellet inference system (Sirin et al. 2007) which is used for reasoning purposes. The task planner was implemented using the Metric-FF (Hoffmann and Nebel 2001) and LPG-td (Gerevini et al. 2004) planners. The algorithm was applied using the Java programming language which is interfaced with Pellet, OWL, and the planners through API. The ontology in the knowledge base was coded by hand, whereas the planning domain and problem were created automatically by Algorithm 4.1 in chapter 4.

In chapter 4 (Figure 4.1), showed the robot environment which is used to test the benefits of employing a semantic action model along with a knowledge base to support the robot task planner. This environment is divided into eight rooms, namely the Kitchen, Office, BedRoom, BathRoom, LivingRoom, Library, DiningRoom and UtilityRoom. Each room is divided into places, which are represented as small squares, to represent the possible locations that can be used by the robot to navigate around its current room or to go out toward other rooms. This environment also contains some objects related to each room, such as a microwave, table, TV, etc. The ontology representation of this environment was shown in Figure 4.2. The following conventions are used in these scenarios: **r** denotes robot; $\mathbf{p_i}$ denotes position i.

[4] http://protege.stanford.edu

7.4.1.1 Knowledge-Based Deterministic Planning and Re-Planning

The aim of this section is to demonstrate the application of the suggested framework and approach presented in Section 7.3. First, the quality of the plan generated by the planning system is evaluated. Then the applicability of the proposed approach to generate alternative plans with missing information related to action parameters is also tested.

Table 7.1 shows the details of several robot actions. The following subsections will explain the scenarios that are used to verify that the task planner can be supported by SAMs and the knowledge base to recover from a missing information scenario.

The environment setting for this scenario follows the same steps as Section 4.6.1 in chapter 4, except that Algorithm 7.1 is used in this scenario to extend the robot initial state and action details.

1. **Bringing Glass of Water Scenario**

   This scenario concerns a user request for the robot to bring him/her a glass of water to the DininingRoom at P14. The goal state of this scenario is therefore: *at*(*robot,P14*) and *on*(*robot_tray, glass_water*). If all the objects which are related to this task are available, then the robot planner will generate an exact plan (with quality = 1) to accomplish this scenario. However, if the robot moves to the kitchen and cannot find a glass of water, what will it do? There are two possible outcomes. The first outcome is that the robot will return a fail, and state that there is no glass of water in the kitchen.

Table 7.1: Robot Actions

No.	Action Name	Inputs	Preconditions	Outputs	Effects
1	Move	robot,from,to	at(Robot,from), isConnected(from,to)	at(robot,to)	not(at(robot,from))
2	Take	robot,object,place	isIn(object,place), at(robot,place), empty(robot-tray)	carry(robot,object)	not(isIn(object,place), not(empty(robot-tray))
3	Drop	robot,object,place	is-droppingposition(place), at(robot,place), carry(robot, object)	dropped(object)	is-in(object, place), not(carry(robot, object), empty(robot-tray)

The second possible outcome is that the robot planner generates an approximate plan according to an accuracy threshold defined by the user.

As explained in Section 7.3, approximate plans are generated by depending on equivalent or similar objects to the ones in the action parameters (in this task the object is a glass of water). Table 7.2 shows the degree of similarities between objects stored in the knowledge base where the calculation of these similarities is based on Figure 7.2. In this case the most similar object to a glass of water is a bottle. Because of the structural hierarchy and inheritance features of the ontology, water inherits liquid features and can then be put in a bottle. The generated plan to solve this task is thus:

take(*r, bottleofwater, p31*),*move* (*r*,*p31*,*p32*),*move* (*r*,*p32*,*p34*),*move* (*r*,*p34*,*p21*),

move (*r, p21, p8*),*move* (*r, p8, p10*),*move* (*r, p10, p13*),*move* (*r, p13, p14*)
drop (*r, bottleofwater, table*).

The quality of this plan according to Equation 7.4 (Wu and Palmer 1994) is (66%).

Table 7.2: Percentage (%) of Object Similarity.

Objects	**Bottle**	**Bowl**	**Spoon**
Glass	66	33	33
Plate	33	66	33

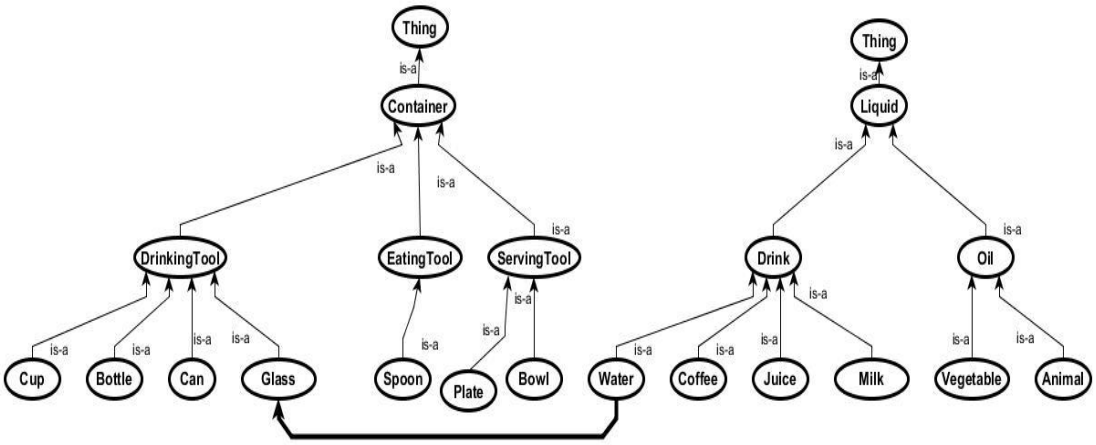


Figure 7.2: Robots can Find Objects Based on their Properties

7.4.1.2 Time Performance of the Planning System under Extended Conditions

The experimental results for deterministic planning times are presented in Table 7.3. These results were obtained from experiments on a 2.50 GHz CORE i5 Intel computer processor at with 8GB RAM. These results' times in milliseconds for pre-planning, transformation of SAMs descriptions in PDDL operators and for the planning operations with LPG-td and Metric-FF. They are acquired after a number of different executions of the system then the average of these results were obtained.

The measurements were applied for domains of different size, i.e. 7, 10 and 13 SAMs. Some of the tests were implemented without semantic refreshment and others were performed with semantic refreshment by using the Wup measurement or the upwards cotopic technique. Table 7.3 shows the ability of the planning system to scale up and efficiently maintain 7, 10 and 13 SAMs.

Pre-planning times did not show significant variation. This is because they depend only on the number and structure of the planning related ontologies, and not on the number of SAMs. Pre-planning hence does not need to be performed again if the domain ontologies are not seriously changed.

The scalability of the system is important because the total transformation time increases as the number of SAMs is increased. However, the estimation of the average transformation time per SAM reveals it to be approximately 2.32 milliseconds for both the exact and the Wup distance metric cases. In the case of the upward cotopic metric distance, the increase in the average transformation time appears to increase as the

number of SAMs is increased. This overhead, which is introduced during the planning system, goes deep in the ontology searching for similar concepts.

By comparing the planning time of Metric-ff and LPG-td, it was found that the planning time in Metric-ff increases as the number of actions increases. However, it is still reasonably fast, as it uses graph structures to exclude unrelated operators early in the planning process. Planning time is not the most important feature affecting system performance, as the planning system is not restricted to a specific planner. Semantic refreshment does not force additional overheads to planning, as it does not increase the number of operators.

Table 7.3: Preplanning Time Measurements in Milliseconds for Experiments in the Planning System.

Number of SAMs		**7**	**10**	**13**
Preplaning Time		14512	15264	15344
Total Transformation Time	**Exact**	387	698	973
	Wup	395	740	995
	Cotopic	489	871	1200
Transformation Time per SAM	**Exact**	55	70	75
	Wup	56	74	77
	Cotopic	70	87	92
Planning Time (Metric-FF)	**Exact**	40	26	28
	Wup	30	32	33
	Cotopic	30	28	26
Planning Time (LPG-td)	**Exact**	26	30	24
	Wup	23	29	23
	Cotopic	20	27	21
Plan Quality	**Exact**	1	1	1
	Wup	0.963	0.878	0.78
	Cotopic	0.74	0.65	0.64

7.4.2 Comparing Time Performance of Planning under Extended Conditions with other Planning Methods

The experimental results in terms of time taken for planning with deterministic, probabilistic and information gathering situations are presented in Table 7.4. These results were obtained from experiments on a 2.50 GHz CORE i5 Intel computer processor with 8GB RAM. The results relate to durations in units of milliseconds for the processes of (i) pre-planning, (ii) transformation of SAMs descriptions in PDDL operators and planning operations with LPG-td and Metric-FF, and acquired after a number of different executions of the system.

Measurements were taken for domains of different sizes, including 7, 10 and 13 SAMs descriptions. Some experiments were conducted without semantic relaxation, i.e. the demand for accurate identification of the inputs and outputs of individual SAMs. These are shown in the table columns labelled as “Fix” or “Pure”. Semantic refreshment is allowed to find approximations of plans, and is labelled as “Sim”, while information gathering under probabilistic conditions is labelled “Prob.”. The labels “Det.Plan”, “Prob.Plan” and “IG.Plan” refer to deterministic, probabilistic and information gathering planning respectively.

It can be seen from Table 7.4 that planning with information gathering takes more time than the other types of planning. This is due to the fact that planning with IG requires the planner to generate a plan more than once. First the normal plan is generated, and then additional plans are generated to gather new information.

Table 7.4: Measurements of Time in Milliseconds for Experiments Conducted in the Planning System for Three Different Situations: Deterministic, Probabilistic and Information Gathering.

Number of SAMs	7					10					13				
System	Det. Plan		Prob. Plan	IG. Plan		Det. Plan		Prob. Plan	IG. Plan		Det. Plan		Prob. Plan	IG. Plan	
	Fix	Sim		Pure	Prob.	Fix	Sim		Pure	Prob.	Fix	Sim		Pure	Prob.
Average Preplaning Time	14612	16300	18945	17523	20845	15274	17840	19542	17942	22750	15542	18235	21861	18687	25231
Average Number of Actions per Plan	10	11	13	11	12	10	13	13	13	14	10	14	14	13	15
Average Total Transformation Time	398	425	497	410	500	411	501	583	456	530	500	598	680	573	693
Average Planning Time (Metric-FF)	25	28	28	25	28	25	30	31	26	32	26	31	3	29	40
Average Planning Time (LGP-td)	20	22	25	21	26	21	23	26	21	28	23	27	30	24	35
Average Number of Plans	1	2	3	2	3	1	2	3	2	3	1	3	3	3	3
Average Plan Quality	1	0.93	0.738	0.878	0.75	1	0.93	0.728	0.821	0.74	1	0.94	0.681	0.82	0.65

7.5 Discussion

The use of semantic knowledge in task planning aids the generation of alternative plans in unexpected situation which might face the robot operation. This situation could be missing some objects which are important in grounding the robot action parameters, i.e., replacing the parameters in the action model with possible values from the semantic knowledge domain. This chapter is tackling this problem by using techniques for estimating the similarity between the original object, which is necessary to ground the operator, and the similar objects which are obtained from the robot environment ontology.

This chapter explains the construction of Semantic Realisation and Refreshment Module (SRRM), which implements the similarity techniques and has the responsibility to extend the robot world state and action models with similar objects. A direct consequence of using SRRM in creating alternative plans is that the process of planning to achieve tasks becomes less computationally demanding. This is due to the task planner not in charge of handling the unexpected situation, but this function is left in a separated unit, i.e. SRRM, to be used when dealing with unexpected cases.

The SRRM is also responsible to estimate the quality of the alternative plans to the original one. The quality of the plan is estimated by depending on the type of the similarity techniques used and the type of the hierarchical relationships between the original object and the new similar objects.

The main contributions of this chapter are:

- The planning system which was proposed in chapter 4 has been developed by adding new module which is called Semantic Realisation and Refreshment Module (SRRM). This module has the ability to calculate the similar objects to the original object in the action by using similarity methods and then extend the robot actions and world states with these similar objects.

- An algorithm, which is based on DL reasoning ability, has been developed. This algorithm has been invoked when no plan can be generated (no exact matching between action inputs in the planning domain and information available in robot world states). Therefore, this algorithm generates an extension to the initial state and the action details by appending semantically equivalent or similar predicates onto both of them.

- The quality of the alternative plans is then tested in order to assess which plan is the best one to select. Assessment methods are used to measure the accuracy of each plan.

7.6 Summary

This chapter tackle the problem of lack of information in the robot world states in the approach which is different from the approach which was presented in chapter 6. So this chapter started by presenting the advantage of using the techniques of measuring the similarity between the original object and other objects. Then a motivation example was presented to explain the necessity of the proposed method in this chapter.

So, the robot planning system can deal with and recover from unexpected situations in which the initial states suffer from missing information. Missing of information causes the task planner fails in generating the suitable plan to solve the robot task. The solution can be obtained by creating alternative plans to the original one by extending the robot initial state and action set with new concepts by using objects that are semantically equivalent or similar to the object in the query. The quality of these new plans is tested in order to select the most suitable plan for the current task.

Chapter 8

Conclusions, Contributions and Future Work

8.1 Conclusions

This thesis has addressed the topic of semantic based symbolic plan generation by autonomous mobile robots acting in domestic environments. The main concentration has been on the ability of a mobile robot's planning system to generate plans and to detect unexpected situations at planning time. The thesis has also discussed the case of recovering from unexpected situations, with a particular focus on uncertainty and lack of information which are required for correct generation of robot task plans.

The successful generation of robot action plans enables the required feature of autonomy to take place. This is a fundamental requirement of mobile robots. Existing symbolic plan generation approaches depend mainly on the results of checking the explicit effects of given actions. Effects encoded in the action model are compared against the evidence database (perception information) to determine whether they are verified or not. These approaches require that the action effects should be clearly noticeable by the robot and this is not always the case in the real world. To deal with this issue, this thesis has proposed to increase the reliability of the process of plan generation by using more developed forms of knowledge representation and reasoning techniques.

A Semantic Knowledge Base (SKB) has been proposed as a source of knowledge to derive implicit expectations about the effects of actions, and to check these expectations using the available information in the evidence database. A new model of

the high-level robot actions has also been developed. This model represents the robot actions as ontology that describes the details of the action, i.e. its parameters, preconditions and effects, by applying semantic knowledge. This model is called a Semantic Action Model (SAM). SKB and SAM have been integrated in order to provide the robot with the ability to be more aware of the semantics of the places and objects in its environment. While a few studies have used semantic knowledge in general robotics applications (Ekvall et al., 2007; Galindo et al., 2005; Bouguerra et al., 2007), this study is the first to use SKB and SAM for the purpose of semantic plan generation.

A general algorithm has been developed for planning systems which are based on robot knowledge ontology in order to control the process of generating a robot plan for a given task. The knowledge ontology is represented by description logics that have the ability to reason about semantic details of objects and places. This provides the planning system with the ability to express general knowledge about the concepts of places and objects in a precise representation. Moreover, the ontology allows the derivation of implicit knowledge from the explicit knowledge. This enables large information sets about the objects' details to be stored as implicit information. In this case the semantic knowledge base will contain less explicit details about objects and places in the robot environment, leaving the implicit details to be inferred by the planning system using techniques of reasoning.

SKB is also used in the case of uncertainty. Uncertainty is a common feature in robotics applications, and therefore a probabilistic planning system to deal with uncertainty in action effects and world states has been developed. The ability to reason about uncertainty endows the planning system with the ability to infer whether

a particular action effect is more likely to be selected given the available evidence information.

The probabilistic planning system considered uncertainty through creating a model that expresses the properties and rules of the semantic knowledge base in a probabilistic way. This model is called a Markov Logic Networks (MLN) and contains the predicates (properties in the SKB), and formulas (rules in the SKB). This model is trained by using the data in the evidence base, allowing it to become a learned model which is ready to answer most of the queries generated by the planning system. The MLN model permits a statement of the probability of finding or not finding an object in the robot's environment. The model also accounts for classifying places and objects which appear in the action details.

The experimental results have shown that semantic knowledge can contribute to support plan generation techniques, especially when uncertainty in the robot world states and action effects are explicitly taken into account. The main objective of this work was to support the robot task planning to generate a semantic plan by evaluating the explicit and implicit effects of plan actions. These effects could consist of the new position of the robot (in the navigation scenario), or the type of object (in the manipulation scenario).

The other objective of this thesis is to enable robots to deal with unexpected situations, so that a robot could continue functioning and complete its task successfully. Coping with unexpected situations can endow the robot planning system with the automation to keep running and to generate alternative plans to accomplish its tasks.

In this thesis, two types of unexpected situations are considered. The first is caused by a lack of information, i.e. incomplete information available in the robot world state. Chapter 6 presented a solution to this problem by using the information gathering method to fill in the blanks where information is lacking. The new information obtained from information gathering techniques is used in two ways. The first is to extend the robot work space with new information in order to support the robot task planning in generating the plan for its current task. The second is used to evaluate the implicit expectations whose truth values cannot be validated by depending on the available world state. In this work a new type of action is developed which has the ability to access the knowledge bases and return new information to robot world space suffering from a lack of information. These actions are called Semantic Action Model for Information Gathering or SAM_IG for short.

The second type of unexpected situation is caused by missing information, e.g. the target object is missing from the robot environment. The planning system dealt with this issue by depending on the concept of similarity between the target object and other objects in the knowledge base. The similarity techniques are used to enrich the robot world state with similar or equivalent objects and to extend the action details (preconditions and effects) with these similar objects. Two types of similarity techniques are used: cotopic similarity and Wup similarity (which are discussed in Chapter 7). The situation of missing information results in alternative plans being generated, each with different qualities. In this case, the quality of each plan is calculated in order to select the most suitable one.

The proposed information gathering method includes steps for modelling the occurring incomplete information as a planning problem. It also includes steps to generate the

planning domain from the SAM_IGs. The planner is then invoked to generate a course of actions which are responsible for collecting the necessary information to solve the lack of information problem. The use of planning to collect new information from the knowledge bases was motivated by its ability to handle complex situations involving a lack of information in an automatic and flexible way.

Although the experiment and results were obtained from simulation, there is no reason that prevents it being applied to a real robot. The work in this thesis is conducted at high levels of robot planning (symbolic level). Study on real robots needs working on low level (actuation and signals).

The planning process is a computationally expensive system, specifically if the planner has to reason about recovering from unexpected and uncertain cases observable in the environment. This study therefore frees the planning from that overhead, and lets it to do the main job, which is to find the plan. The computational demands of reasoning, extending the robot work space with new information and information gathering methods are allocated to separate units such as SRRM, MLN and IG.

8.2 Contributions

The main contributions of the work reported in this thesis are in the area of plan generation in mobile robotics. These contributions are:

- The concept of using the Semantic Knowledge Base (SKB) to support the robot task planner under deterministic conditions is defined. A new model of

high-level robot actions has been defined, and this model represents the details of robot action as ontology. This model is thus known as the Semantic Action Model (SAM). An algorithm that integrates SKB and SAMs has also been developed. This algorithm creates the planning domain in the Planning Domain Definition Language (PDDL) style. This is used as input to the planner to generate the plan for robot tasks. Then, a general purpose planning algorithm has also been defined, which can support planning under deterministic conditions, and is based on using ontology to represent the semantic knowledge base. Although the use of a semantic knowledge base is becoming more commonplace in some mobile robotic areas (such as mapping and human robot interaction), it is virtually non-existent in details in plan generation. This work is the first to propose the use of SKB and SAM to generate semantic robot plans, and that area is therefore considered to be a major contribution of this thesis. Chapter 4 has presented the aforementioned contribution. It describes (a) SKB, (b) SAMs, (c) an algorithm for generating PDDL from SKB and SAM, and (e) general purpose task planning algorithm.

- A further contribution relates to the development of a probabilistic approach to deal with uncertainty in semantic knowledge based task planning. Chapter 5 has presented the probabilistic approach and discusses how uncertainties in action effects and world states are taken into account by the planning process. This contribution also served to resolve situations of confusion in finding an object relevant to the successful generation of an action during task planning. The accuracy related to this type of planning, in average, is (90.10%) in navigation scenario.

- An additional contribution is the study of using the planning system to respond to unexpected situations which are caused by lack of information. This contribution is formalised as a general approach that models events of incomplete knowledge as a planning problem. This result in developing a new type of action which is known as a Semantic Action Model for Information Gathering (SAM_IG). These actions have the ability to access the knowledge base to retrieve the necessary information to support the planning system when it is faced with incomplete information. The information gathering approach is also used to gather the necessary information in order to check the implicit expectations of the generated actions. The correct classification related to this type of planning in navigation scenario in average is (92.83 %). Chapter 6 has presented this approach.
- Another contribution is the study of using semantic similarity methods to support the planning system to recover from situation of missing information (target) problem. This result in developing Semantic Realisation and Refreshment Module (SRRM) which has the ability to estimate the similarities between objects in order to extend the robot world states with new similar objects to the original object in the action model. Also, the SRRM has been used to estimate the quality of the alternative plans. The quality of the alternative plans could be similar to the original plan, in average, by 93%. Chapter 7 has presented this approach.

8.3 Limitations

The semantic plan generation, which is based on the semantic knowledge domain, is applicable on domains that have expressive semantic information available. This means that the knowledge base should have information about how its objects are grouped into classes and how the properties between those classes are structured and restricted.

The limitation in the semantic based planning system approach is the choosing of the object within the action that has to check its details constraints, i.e. its implicit information, against the available evidence data during plan generation. In the proposed planning system, the process of checking begins with the selection of objects in the add list of action effects (add(o)).

Then all the objects' implicit details should be verified. This process is well if the number of objects available in the positive effect part of the action is low. But if the number of objects is large then the process may be difficult in terms of computational time and memory resources. For example, in the deterministic planning system, the action *move(robot,hall,r1)* has the effect *at(robot,r1)*, which explains that the robot should end up in room *r1*. The object to be checked in this case is *r1*. This process is fine for a small number of objects related to *r1*, but becomes difficult for higher numbers of objects. This issue hence warrants further investigation in order to find alternative approaches to dealing with it in the planning system.

8.4 Future Work

There were a few cases that required deeper exploration in order to improve the proposed planning system such that this system be better exploit the semantic domain knowledge. These cases are described below.

In the probabilistic planning system process, the MLN models are completely dependent upon the knowledge available within the SKB. This knowledge base also contains the explicit definitions of the classes and relationships between these classes. The implicit information of the objects inside the SKB is obtained by using reasoning techniques, so implicit information, according to the proposed planning system, cannot be engaged in the created MLN. This issue needs further investigation in order to add the implicit information into the process of creating an MLN. Another issue in the MLN model is related to the time required for the learning algorithm to learn the MLN. It can take varying degrees of time to learn this model. Hence enhancing the learning time of the MLN should also be considered in future work.

Regarding the proposed planning system that deals with situations which lack some information, there are two open issues that need to be addressed. The first one is related to the interaction between the main planning task and the information gathering task. In the current implementation, information gathering tasks are not allowed to modify the world state of the robot that was reached by the planner. This is a severely limiting condition that might hinder the information gathering plan in completing its task. For this reason, further investigation is needed to find an approach which coordinates the generation of both plans so that the assigned task is achieved successfully.

The second issue is that the situations involving a lack of information might need to validate a large number of implicit expectations which have ambiguous values. This issue complicates the information gathering approach. One possible solution to this issue would be to identify only a small set of expectations to be checked, then generate a plan to collect information related to them. This set of expectations should be more suitable for solving the problem of incomplete information.

Integrating knowledge with the task planner is an interesting pursuit within the field of Artificial Intelligence and robotics applications. This thesis has attempted the challenge of exploring and developing a high-performance robot task planner that integrates deterministic and uncertain operations within the planning system. However, much remains to be achieved before AI systems can reach human levels of intelligence. A powerful knowledge representation system can lead to more facilities to deal with unexpected situations that might face a robot in the real world. The knowledge ontology that is stored as descriptive logics in OWL format provides flexibility in the planning system when generating its plan. This thesis has also made a concerted effort to build powerful algorithms for dealing with tasks that might be deterministic or suffered from missing, incomplete or uncertain information.

Appendix

Appendix A

A.1 Semantic Knowledge Base

A.1.1 Navigation Domain

Properties

isContained :domain *Room* :range *Object*

isLocatedAt :domain *Object* :range *Room*

isInside :domain *Object* :range *Container*

hasObject :domain *Place* :range Object

Atomic Classes

Room	*Oven*	*Tv-Set*	*WashingMachine*
Object	*Bed*	*Sink*	*Pc*
Place	*Sofa*	*Chair*	*Iron*
Hall	*Table*	*Tub*	*DiningTable*

Defined Classes

BathRoom

and (*isContained* only *BathRoomObject*) and (*isContained* max 0 (*BedRoomObject* or *DiningRoomObject* or *KitchenObject* or *LibraryObject* or *LivingRoomObject* or *OfficeObject* or *UtilityRoomObject*))

Kitchen

and (*isContained* only *KitchenObject*) and (*isContained* max 0 (*BathRoomObject* or *BedRoomObject* or *DiningRoomObject* or *LibraryObject* or *LivingRoomObject*

or *OfficeObject* or *UtilityRoomObject*))

LivingRoom

and (*isContained* only *LivingRoomObject*) and (*isContained* max 0 (*BathRoomObject*

or *BedRoomObject* or *DiningRoomObject* or *KitchenObject* or *LibraryObject*

or *OfficeObject* or *UtilityRoomObject*))

DiningRoomObject

and (*isLocatedAt* only *DiningRoom*) and (*isLocatedAt* max 0 (*BathRoom* or *BedRoom*

or *Kitchen* or *Library* or *LivingRoom* or *Office* or *UtilityRoom*))

LibraryObject

and (*isLocatedAt* only *Library*) and (*isLocatedAt* max 0 (*BathRoom* or *BedRoom* or *DiningRoom* or *Kitchen* or *LivingRoom* or *Office* or *UtilityRoom*))

A.1.2 Manipulation Domain

Properties

hasHandle :domain *Cup* :range *Handle*

hasCover :domain *Container* :range *Cover*

hasCap :domain *Bottle* :range *Cap*

Atomic Classes

Handle

Cover

Cap

Container

Defined Classes

Cup

and (*isA Container)* and (*hasHandle* only *Handle*) and (*hasCover* max 0 *Cover*) and (*hasCap* max 0 *Cap*)

GlassofWater

and (*isA Container)* and (*hasHandle* max 0 *Handle*) and (*hasCover* max 0 *Cover*) and (*hasCap* max 0 *Cap*)

Bottle

and (*isA Container)* and (*hasCap* max 1 *Cap*) and (*hasHandle* max 0 *Handle*) and (*hasCover* max 0 *Cover*)

Bowl

and (*isA Container)* and (*hasCap* max 0 *Cap*) and (*hasHandle* max 0 *Handle*) and (*hasCover* max 0 *Cover*)

A.2 Evidence Base

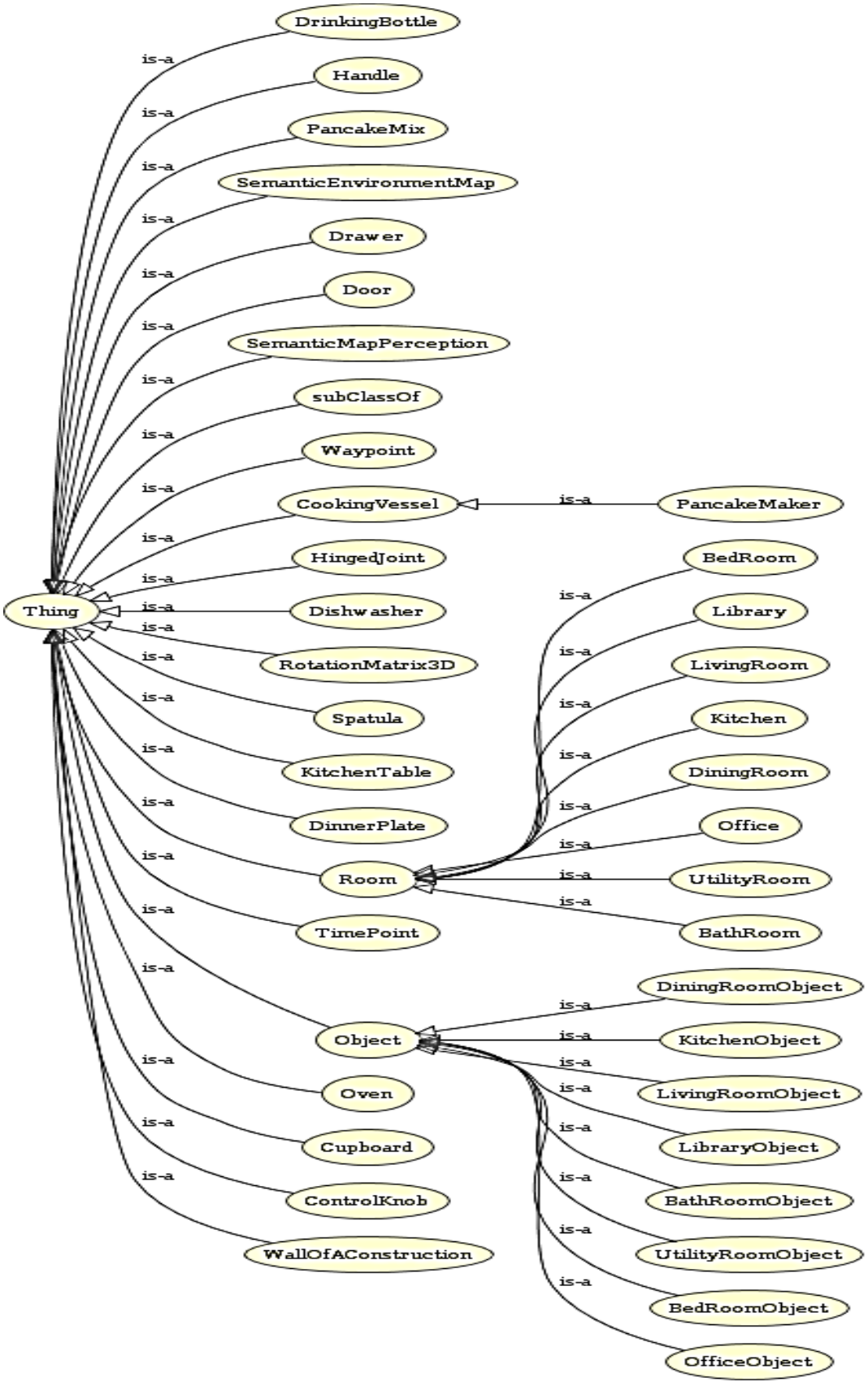


Figure A.1: Evidence Base

A.3 Robot Planning Domain

(define (domain experiment)

(:requirements :typing :fluents)

(:types location object robot tray)

(:constants robot - object)

(:predicates

(at ?robot – robot ?loc - location)

(free ?obj - object)

(on ?obj1 - object ?obj2 - object)

(in ?obj1 - object ?obj2 - object)

(free-robot - robot)

(taken ?obj - object)

(closed ?obj - object)

(opened ?obj - object)

(can-move ?robot - robot ?x - location ?y - location)

(charge-station ?x - location)

(isconnected ?x - location ?y - location)

(empty?robot- tray)

(carry ?robot-robot ?obj-object)

(is-ropping-position ?loc-location)

(dropped ?obj-object)

```
(is-in ?obj - object ?loc - location)

)

(:functions

(time-to-move ?l1 ?l2 - location)

 (time)

)

(:action move

:parameters

 (?robot - robot ?loc-from - location ?loc-to - location

)

:precondition

(and (at ?robot - robot ?loc-from) (can-move ? robot - robot ?loc-from ?loc-to)

(isconnected ?loc-from ?loc-to) (isconnected ?loc-to ?loc-from)

)

:effect

(and (not (at robot - robot ?loc-from)) (at ?robot-robt ?loc-to)

(increase (time) (time-to-move ?loc-from ?loc-to))

)

)


 (:action takeout
```

:parameters

(?obj - object ?obj2 - object ?loc - object

)

:precondition

(and (at robot - robot ?loc - location) (at ?obj2 - object ?loc - location) (in ?obj - object ?obj2 - object) (opened ?obj2 - object) (free-robot - robot)

)

:effect

(and (not (free-robot - robot)) (not (at ?obj - object ?loc - location)) (taken ?obj - object) (not (in ?obj - object ?obj2 - object)))

)

(:action take

:parameters

(?robot – robot ?obj - object ?loc - location

)

:precondition

(and (at robot – robot ?loc - location) (is-in ?obj - object ?loc - location)

(free ?obj - object) (free-robot - robot) (empty ?tray-tray)

)

:effect

(and (not (free ?obj - object)) (not (free-robot - robot)) (not (at ?obj - object ?loc - location)) (carry ?robot-robot obj-object) (taken ?obj - object) (not) (is-in ?obj - object ?loc - location)) (not(empty ?tray-tray)))

)

(:action remove

:parameters

(

?obj - object ?obj2 - object ?loc - location

)

:precondition

(and (at ? robot - robot ?loc - location)

(at ?obj - object ?loc - location)

(on ?obj1 - object ?obj2 - object)

(free ?obj - object)

(free-robot - robot)

)

:effect

(and (not (free ?obj - object)) (not (free-robot - robot)) (not (at ?obj - object ?loc - location)) (taken ?obj - object))

)

(:action give

:parameters

(?obj - object ?loc - location

)

:precondition

(and (at ? robot - robot ?loc - location) (taken ?obj - object)

)

:effect

(and (not (taken ?obj - object)) (at ?obj - object ?loc - location) (free-robot ? robot - robot))

)

(:action putdown

:parameters

(?obj1 - object ?obj2 - object ?loc - location

)

:precondition

(and (at ? robot - robot ?loc - location) (at ?obj1?loc - location) (at ?obj2 - object ?loc - location) (on ?obj1- object ?obj2 - object) (free ?obj1 - object))

:effect

(and (not (on ?obj1 - object ?obj2 - object)) (free ?obj - object))

)

(:action drop

:parameters

?robot-robot ?object-object ?place-location

:precondition

(is-droppingposition ?place-location) (at robot-robot ?place-location) (carry ?robot-robot ?object-object)

:effect

(dropped ?object-object) (is-in ?object-object ?place-location) (not(carry ?robot-robot ?object-object) (empty ?robot-tray-tray)

)

)

References

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

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

American Psychological Association. (2005). *Concise Rules of APA Style*. Washington, DC: APA Publications.

Anderson, M.L. 2003. Embodied Cognition: A field guide. *Artificial Intelligence* 149(1), pp. 91–130.

Baader, F., Horrocks, I. and Sattler, U. 2008. Description Logics. In: Harmelen, F. van et al. eds. *Handbook of Knowledge Representation*. 1st ed. Elsevier, pp. 135–179.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P.F. 2010. *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd ed. Cambridge University Press.

Bacchus, F. 2001. AIPS 2000 Planning Competition. *AI Magazine* 22(3), p. 47.

Bechhofer, S. Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., and Lynn, Andrea S. 2004. OWL Web Ontology Language Reference [Online] Available at: http://www.w3.org/TR/owl-ref/ [Accessed: 15 December 2013].

Beetz, M. Stulp F., Esden-Tempski, P., Fedrizzi, A., Klank, U., Kresse, I., Maldonado, A. and Ruiz, F. 2009. Generality and legibility in mobile manipulation. *Autonomous Robots* 28(1), pp. 21–44.

Berger, J.O. 1985. *Statistical Decision Theory and Bayesian Analysis*. 2nd ed. New York: Springer-Verlag.

Berners-Lee, T., Hendler, J., and Lassila, O. 2001. The Semantic Web. *Scientific American* 284(5), pp. 34–43.

Bickhard, M.H. and Terveen, L. 1995. *Foundational issues in artificial intelligence and cognitive science*. Elsevier Science.

Blodow, N., Goron, L. C., Marton, Z., Pangercic, D., Ruhr, T., Tenorth, M., and Beetz, M. 2011. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4263–4270.

Blum, A.L. and Furst, M.L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2), pp. 281–300.

Bohren, J., Rusu, R. B., Jones, E. G., Marder-Eppstein, E., Pantofaru, C., Wise, M., Mosenlechner, L., Meeussen, W. and Holzer, S. 2011. Towards autonomous robotic butlers: Lessons learned with the PR2. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 5568–5575.

Bonet, B., Loerincs, G., and Geffiner, H. 1997. A robust and fast action selection mechanism for planning. In: *in Proceedings of the 14th International Conference of the American Association of Artificial Intelligence*. Providence, Rhode Island, pp. 714–719.

Bonet, B. and Geffner, H. 1999. Planning as Heuristic Search: New Results. In: *Proceedings of the 5th European Conference on Planning*. Durham, UK.

Bouguerra, A. Karlsson, L. and Saffiotti, A. 2007a. Handling uncertainty in semantic-knowledge based execution monitoring. In: *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Diego, CA, USA, pp. 437–443.

Bouguerra, A. Karlsson, L. and Saffiotti, A. 2007b. Semantic Knowledge-Based Execution Monitoring for Mobile Robots. In: *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, pp. 3693–3698.

Bouguerra, A. and Karlsson, L. 2004. Hierarchical Task Planning under Uncertainty. In: *In 3rd Italian Workshop on Planning and Scheduling*. Perugia, Italy.

Brooks, R.A. 1990. Elephants don't play chess. *Robotics and Autonomous Systems* 6(1-2), pp. 3–15.

Brooks, R.A. 1991. Intelligence without representation. *Artificial Intelligence* 47(1-3), pp. 139–159.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence* 69, pp. 165–204.

Chan, K.S.M., Bishop, J. and Baresi, L. 2007. *Survey and Comparison of Planning Techniques for Web Services Composition*. South Africa.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3), pp. 333–377.

Chella, A., Cossentino, M., Pirrone, R. and Ruisi, A. 2002. Modeling ontologies for robotic environments. In: *Proceedings of the 14th international conference on Software engineering and knowledge engineering - SEKE '02*. New York, New York, USA: ACM Press, p. 77.

Coradeschi, S. and Saffiotti, A. 2003. An introduction to the anchoring problem. *Robotics and Autonomous Systems* 43(2-3), pp. 85–96.

Deacon, T.W. 1998. *The Symbolic Species: The Co-evolution of Language and the Brain*. W. W. Norton & Company.

Degroot, M. 2004. *Optimal Statistical Decisions*. Wiley-Blackwell; WCL Edition edition.

Devyver, P.A. and Kittler, J. 1982. *Pattern recognition: A statistical approach*. Prentice-Hall.

Duda, R.O., Hart, P.E., and Stork, D.G. 2000. *Pattern Classification*. 2nd ed. New York: Wiley.

Edelkamp, S. and Hoffmann, J. 2004. PDDL 2.2: The Language for the Classical Part of IPC-4. In: *in Proceedings of the International Planning Competition International Conference on Automated Planning and Scheduling (Whistler 2004)*. pp. 1–7.

Eich, M. and Goldhoorn, M. 2010. Semantic Labeling: Classification of 3D Entities Based on Spatial Feature Descriptors. In: *IEEE International Conference on Robotics and Automation (ICRA 2010)*.

Ekvall, S., Kragic, D. and Jensfelt, P. 2007. Object detection and mapping for service robot tasks. *Robotica* 25(2), pp. 175–187.

Erol, K., Hendler, J. and Nau, D.S.,. 1994. HTN Planning: Complexity and Expressivity. In: *In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. pp. 1123–1128.

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

Fikes, R.E. and Nilsson, N.J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3-4), pp. 189–208.

Fisher, A. 2015. Reporting t-tests in APA Style [Online] Available at: www.simplypsychology.org/t-test.html [Accessed: 13 June 2015].

Fox, M. and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, pp. 61–124.

Fujii, T. and Ura, T. 1996. Development of an autonomous underwater robot ?Twin-Burger? for testing intelligent behaviors in realistic environments. *Autonomous Robots* 3(2-3), pp. 285–296.

Galindo, C., Fernandez-Madrigal, J.A. and Gonzalez, J. 2004. Improving Efficiency in Mobile Robot Task Planning Through World Abstraction. *IEEE Transactions on Robotics* 20(4), pp. 677–690.

Galindo, C., González, J. and Fernández-Madrigal, J.A. 2004. Interactive Task Planning through Multiple Abstraction: Application to Assistant Robotics. In: *In 16th European Conference on Artificial Intelligece*. Valencia, Spain.

Galindo, C., Fernandez-Madrigal, J.A., Gonzalez, J., Saffiotti, A., and Buschka, P. 2007. Life-Long Optimization of the Symbolic Model of Indoor Environments for a Mobile Robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 37(5), pp. 1290–1304.

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

Galindo, C., Fernández-Madrigal, J.A., González, J., and Saffiotti, A. 2008. Robot task planning using semantic maps. *Robotics and Autonomous Systems* 56(11), pp. 955–966.

Galindo, C, Fernandez-Madrigal, J.A., Gonzalez, J., and Saffiotti, A. 2007. Using Semantic Information for Improving Efficiency of Robot Task Planning. In: *Proceeding of the ICRA-07 Workshop on Semantic Information in Robotics*. Rome, Italy.

Galindo, C. and Saffiotti, A. 2013. Inferring robot goals from violations of semantic knowledge. *Robotics and Autonomous Systems* 61(10), pp. 1131–1143.

Gerevini, A., Saetti, A., Serina, I. and Toninelli, P. 2004. LPG-TD: a fully automated planner for PDDL2.2 domains. In: *14th International Conference on Automated Planning and Scheduling (ICAPS-04) International Planning Competition abstracts*. Whistler, Canada.

Gerevini, A., Saetti, A., and Serina, I. 2003. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *Journal of Artificial Intelligence Research* 20, pp. 239–290.

Gerevini, A. and Long, D. 2005. *Plan Constraints and Preferences in PDDL3*. Italy.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D. and Wilkins, D. 1998. *PDDL -- the Planning Domain Definition Language*. New Haven, CT.

Giunchiglia, F. and Traverso, P. 2000. Planning as Model Checking. In: *In the Proceedings of 5th European Conference on Planning, ECP'99*. Durham, UK: Springer Berlin Heidelberg, pp. 1–20.

Gupta, R. and Kochenderfer, M.J. 2004. Common sense data acquisition for indoor mobile robots. In: *Proceedings - Nineteenth National Conference on Artificial Intelligence (AAAI-2004): Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-2004)*. San Jose, CA, pp. 605–610.

Haigh, K.Z. and Veloso, M.M. 1997. High-level planning and low-level execution. In: *Proceedings of the first international conference on Autonomous agents - AGENTS '97*. New York, New York, USA: ACM Press, pp. 363–370.

Hamilton, A.G. 1978. *Logic for Mathematicians*. Cambridge, UK: Cambridge University Press.

Hariri, B.B., Abolhassani, H. and Khodaei, A. 2006. A new Structural Similarity Measure for Ontology Alignment. In: *Proceedings of the International Conference on Semantic Web and Web Services*. Las Vegas, USA, pp. 36–42.

Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D., and Vlahavas, I. 2012. An Integrated Approach to Automated Semantic Web Service Composition through Planning. *IEEE Transactions on Services Computing* 5(3), pp. 319–332.

Hauser, L. 1997. Selmer Bringsjord, What Robots Can and Can't Be, Studies in Cognitive Systems. *Minds and Machines* 7(3), pp. 433–438.

Hoffmann, J. 2000. A Heuristic for Domain Independent Planning and its Use in an Enforced Hill-Climbing Algorithm. In: *12th International Symposium, ISMIS 2000 Charlotte, NC, USA, October 11–14, Proceedings*. Charlotte, USA, pp. 216–227.

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

Hois, J., Wünstel, M., Bateman, J. A. and Röfer, T. 2006. Dialog-Based 3D-Image Recognition Using a Domain Ontology. *Spatial Cognition V: Reasoning, Action, Interaction, International Conference Spatial Cognition 2006*, pp. 107–126.

IPC 2004. International Planning Competition 2004 [Online] Available at: http://ipc.icaps-conference.org/ [Accessed: 10 August 2014].

Jain, D., Waldherr, S. and Beetz, M.. 2009. *Bayesian logic networks*. Intelligent Autonomous Systems Group, Technische Universität München, Technical Report.

Jain, D. 2011a. Knowledge engineering with markov logic networks: A review. *Evolving Knowledge in Theory and Applications* , page 16.

Jain, D. 2011b. ProbCog Toolbox [Online] Available at: http://ias.in.tum.de/software/probcog [Accessed: 21 March 2014].

Kautz, H. and Selman, B. 1998. BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In: *In Proceedings of the AIPS-98 Workshop on Planning as Combinatorial Search*. Pittsburgh, Pennsylvania, pp. 58–60.

Kautz, H. and Selman, B. 1992. Planning as Satisfiability. In: *in Proceedings of the 10th European Conference on Artificial Intelligence*. Vienna, Austria, pp. 359–363.

Kautz, H. and Selman, B. 1996. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In: *In Proceedings of the 13th National Conference on Artificial Intelligence*. Portland, Oregon, pp. 1194–1201.

Kemke, C. and Walker, E. 2006. Planning with Action Abstraction and Plan Decomposition Hierarchies. In: *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. IEEE, pp. 447–451.

Knoblock, C.A. and Yang, Q. 1993. A Comparison of the SNLP and TWEAK Planning Algorithms. In: *In Working Notes of the AAAI Spring Symposium Series: Foundations of Automatic Planning: The Classical Approach and Beyond*. pp. 73–77.

Koehler, J., Nebel, B., Hoffmann, J. and Dimopoulos, Y. 1997. Extending planning graphs to an ADL subset. In: *In Proceedings of the 4th European Conference on Planning*. Toulouse, France., pp. 273–285.

Kok, S. and Domingos, P. 2005. Learning the structure of Markov logic networks. In: *Proceedings of 22nd International Conference on Machine Learning*. ACM Press, pp. 441–448.

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

Laborie, P. and Ghallab, M. 1995. IxTeT: an integrated approach for plan generation and scheduling. In: *Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95*. IEEE Comput. Soc. Press, pp. 485–495.

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

Lenat, D.B. 1995. CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11), pp. 33–38.

Lim, G.H., Hwang, W., Suh, H.W. W., Choi, J.H. and Park, Y.T. 2007. Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 429–436.

Liu, H. and Singh, P. 2004. ConceptNet: A Practical Commonsense Reasoning Toolkit. *BT Technology Journal* (22), pp. 211–226.

Long, D. and Fox, M. 1999. Efficient Implementation of the Plan Graph in STAN. *Journal of Arti cial Intelligence Research* 10, pp. 87–115.

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

Martin, D., Ankolekar, A., Burstein, M., Denker, G., Elenius, D., Hobbs, J., Kagal, L.,Lassila, O., McDermott, D., McGuinness, D., McIlraith, S., Paolucci, M., Parsia, B.,Payne, T., Sabou, M., Schlenoff, C., Sirin, E., Solanki, M., Srinivasan, N., Sycara, K. and Washington, R. 2004. OWL-S 1.1 Release [Online] Available at: http://www.daml.org/services/owl-s/1.1/ [Accessed: 15 March 2013].

Mcallester, D. and Rosenblitt, D. 1991. Systematic Nonlinear Planning. In: *In Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*. Anaheim, California, pp. 634–639.

Mcdermott, D. 1996. A Heuristic Estimator for Means-Ends Analysis in Planning. In: *In Proceedind of International Conference on AI Planning Systems (AIPS-96)*. pp. 142–149.

McDermott, D. 2002. Estimated-Regression Planning for Interactions with Web Services. In: *In Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems*. AAAI Press.

McDermott, D. 2005. Opt manual [Online] Available at: http://cs-www.cs.yale.edu/homes/dvm/papers/opt-manual.pdf.

Mcdermott, D. 2000. The 1998 AI Planning Systems Competition. *AI Magazine* 21, pp. 35–55.

McDermott, D. 1999. Using regression-match graphs to control search in planning. *Artificial Intelligence* 109(1-2), pp. 111–159.

Mitchell, T.M. 1997. *Machine Learning*. Internatio. New York: McGraw-Hill Higher Education.

Motik, B., Patel-Schneider, P.F., Parsia, B., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U. and Smith, M. 2009. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) [Online] Available at: http://www.w3.org/TR/owl2-syntax/ [Accessed: 10 March 2015].

Mozos, O., Jensfelt, P., Zender, H., Kruijff, G. and Burgard, W. 2007. From Labels to Semantics: An Integrated System for Conceptual Spatial Representations of Indoor Environments for Mobile Robots. In: *Proceedings of the IEEE ICRA Workshop: Semantic information in robotics*. pp. 33–40.

Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In: Fox, M. S. and Zweben, M. eds. *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufmann, pp. 169–212.

Muscettola, N., Nayak, P.P., Pell, B. and Williams, B. C. 1998. Remote Agent: to boldly go where no AI system has gone before. *Artificial Intelligence* 103(1-2), pp. 5–47.

Newell, A. and Simon, H.A. 1976. Computer science as empirical inquiry: symbols and search. *Communications of the ACM* 19(3), pp. 113–126.

Newell, A. and Simon, H.A. 1963. *GPS, a program that simulates human thought*. New York: McGraw-Hill.

Nualláin, S.Ó., Mc Kevitt, P. and Aogáin, E. M. 1997. *Two Sciences of Mind: Readings in cognitive science and consciousness*. John Benjamins Publishing.

Nüchter, A., Wulf, O., Lingemann, K., Hertzberg, J., Wagner, B. and Hartmut, S. 2006. 3D Mapping with Semantic Knowledge. In: *RoboCup 2005: Robot Soccer World Cup IX*. pp. 335–346.

O'Kane, J.M., Tovar, B. and Cheng, P. 2006. Algorithms for planning under uncertainty in prediction and sensing. In: *Series in Control Engineering*. Marcel Dekker.

Ong, S.C.W., Shao, W. P., Hsu, D. and Wee, S. L. 2010. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research* 29(8), pp. 1053–1068.

Papadimitriou, C.H. and Tsitsiklis, J.N. 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research* 12(3), pp. 441–450.

Penberthy, S.J. and Weld, D.S. 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In: *In Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. pp. 103–114.

Poon, H. and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In: *Proceedings of the 21st national conference on Artificial intelligence (AAAI)*. AAAI Press, pp. 458–463.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng A. 2009. ROS: an open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*. pp. 1–9.

Reiser, U. , Connette, C., Fischer, J., Kubacki, J., Bubeck, A., Weisshardt, F., Jacobs, T., Parlitz, C., Hagele, M. and Verl, A. 2009. Care-O-bot® 3 - creating a product vision for service robot applications by integrating design and technology. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1992–1998.

Richardson, M. and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2), pp. 107–136.

Sacerdott, E.D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5(2), pp. 115–135.

Saffiotti, A. and Broxvall, M. 2005. PEIS Ecologies: Ambient intelligence meets autonomous robotics. In: *Proceeding of the International Conference on Smart Objects and Ambient Intelligence (sOc-EUSAI)*. pp. 275–280.

Shannon, C.E. 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5(1), p. 3.

Singla, P. and Domingos, P. 2005. Discriminative training of Markov logic networks. In: *Proceedings of the 20th national conference on Artificial intelligence (AAAI)*. AAAI Press, pp. 868–873.

Sirin, E. , Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2), pp. 51–53.

Srinivasa, S.S., Ferguson, D., Helfrich, C. J., Berenson, D., Collet, A., Diankov, R., Gallagher, G., Hollinger, G., Kuffner, J. and Weghe, M. V. 2009. HERB: a home exploring robotic butler. *Autonomous Robots* 28(1), pp. 5–20.

Takeuchi, I. and Furuhashi, T. 1998. Self-organisation of grounded symbols for fusions of symbolic processing and parallel distributed processing. In: *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence*. IEEE, pp. 715–720.

Taniguchi, T. and Sawaragi, T. 2004. Self-organization of inner symbols for chase: symbol organization and embodiment. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. IEEE, pp. 2073–2079.

Tenorth, M., Jain, D. and Beetz, M. 2010. Knowledge Representation for Cognitive Robots. *In Künstliche Intelligenz, Springer* 24(3), pp. 233–240.

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

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

Tenorth, M. and Beetz, M. 2013a. Exchanging Action-related Information among Autonomous Robots. In: *Intelligent Autonomous Systems 12*. Springer, pp. 467–476.

Tenorth, M. and Beetz, M. 2009. KNOWROB — knowledge processing for autonomous personal robots. In: *EEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4261–4266.

Tenorth, M. and Beetz, M. 2013b. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research (IJRR)* 32(5), pp. 566–590.

Theobalt, C., Bos, J., Chapman, T., Espinosa-Romero, A., Fraser, M., Hayes, G., Klein, E., Oka, T. and Reeve R. 2002. Talking to Godot: dialogue with a mobile robot. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1338–1343.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, Ch., Roy, N., Schulte, J. and Schulz, D. 1999. MINERVA: a second-generation museum tour-guide robot. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. IEEE, pp. 1999–2005.

Veloso, M.V., Carbonell, J., Pérez, A., Borrajo, D. and Blythe, J. 1995. Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Experimental and Theoretical Artificial Intelligence* 7, pp. 81–120.

Vrakas, D. Tsoumakas, G., Bassiliades, N. and Vlahavas, I. 2005. HAPrc: An Automatically Configurable Planning System. *AI Communications* 18(1), pp. 41–60.

Vrakas, D, Refanidis, I., Milcent, F. and Vlahavas, I. 1999. On the Parallelization of Greedy Regression Tables. In: *In Proceedings of the 18th Workshop of the UK Planning and Scheduling Special Interest Group*. Manchester, UK, pp. 180–189.

Wei, W., Erenrich, J., Selman, B. 2004. Towards Efficient Sampling: Exploiting Random Walk Strategies. In: *Nineteenth National Conference on Artificial Intelligence*. AAAI Press, pp. 670–676.

Wu, Z. and Palmer, M. 1994. Verbs Semantics and Lexical Selection. In: *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 133–138.

Yang, Q. and Tenenberg, J.D. 1990. ABTWEAK: Abstracting a nonlinear, least commitment planner. In: *In Proceedings of the Eighth National Conference on Artificial Intelligence*. pp. 204–209.