Universidad Carlos III de Madrid

Escuela Politécnica Superior

Programa de Doctorado en Ciencia y Tecnología Informática

Thesis Proposal

Automated Planning Through Abstractions in Real World Environments

Author: Moisés Martínez

Advisors: Daniel Borrajo and Fernando Fernández

Contents

List of Figures

List of Tables

Abstract

Generating sequences of actions – plans – for an automatic system, like a robot, using Automated Planning is particularly difficult in stochastic and/or dynamic environments. These plans are composed of actions whose execution, in certain scenarios, might fail, which in turn prevents the execution of the rest of the actions in the plan. Also, in some environments, plans must be generated fast, both at the start of the execution and after every execution failure. These problems have contributed to generate new Automated Planning models (Planning under uncertainty) to tackle these situations. These models include changes in the representation of the information to manage the dynamics of the environment (action outcomes, observability of the environment, etc). In spite of the initial advantages of these models, there are some important disadvantages that increase the cost of generating a plan. These models require an accurate definition of the environment's dynamics. Frequently, it is extremely difficult to define such accurate models, and in some environments the amount of information needed is huge. The most common solution to avoid these problems consists on repairing or re-planning when a failure in execution is detected due the lack of information. Therefore, at each planning (re-planning) step, a new plan of actions is generated including the possible changes in the environment. The goal of this thesis is to define a new planning approach that allows to reduce the computational effort of the planning task in real world environments, even adding new information discovered on execution. This thesis proposes the use of abstractions that focus on the information that is related with the current state. This technique generates abstract plans where the sequence of the first actions in the plan is guaranteed to be applicable as long as the information about the environment is complete, the actions' execution is correct and the environment does not change, while the rest of the plan might not necessarily be applicable.

Resumen

Generar secuencias de acciones – planes – para un sistema automático, como un robot, mediante la utilización de Planificación Automática es particularmente difícil en entornos estocásticos y/o dinámicos. Estos planes están compuestos por acciones cuya ejecución puede fallar en algunas ocasiones, evitando que se puedan ejecutar el resto de acciones que componen el plan. Además, en algunos entornos, los planes deben ser generados rápidamente, tanto al comienzo de la ejecución, como cuando un fallo es detectado durante ésta. Estos problemas han contribuido a que aparezcan nuevos modelos de Planificación Automática (Planificación con incertidumbre) capaces de manejar estos problemas. Estos modelos incluyen cambios en la representación de la información para gestionar las dinámicas del entorno (resultados de las acciones, observabilidad del entorno, etc). A pesar de la ventajas iniciales de estos modelos, existen algunas importantes desventajas que producen un incremento en el coste de generación de los planes de acciones. Esto es debido a que es necesario tener una representación precisa de la dinámica del entorno y frecuentemente es extremadamente complicado obtenerla o es demasiado grande para manejarla. La solución más común para evitar estos problemas consiste en reparar o re-planificar cuando se detecta un fallo durante la ejecución debido a la falta de información. Por lo tanto, en cada proceso de planificación (re-planificación), se genera un nuevo plan de acciones incluyendo los posibles cambios detectados en el entorno. El objetivo de esta tesis es definir una nueva técnica que permita reducir la complejidad de la tarea de Planificación Automática en entornos reales con incertidumbre, incluso añadiendo nueva información que es descubierta durante el proceso de ejecución. De forma más precisa, en esta tesis se propone el uso de abstracciones que permitan centrar el proceso de planificación en la información más relacionada con el estado actual. Esta técnica genera un plan abstracto donde el primer conjunto de acciones pueden ser ejecutadas si la información acerca del entorno es completa, la ejecución de la acciones es correcta y el entorno no cambia, mientras que el resto del plan puede no ser aplicable.

Chapter 1

Introduction

In the last decade, there was a growing need to build control systems with the ability to interact in complex stochastic and dynamic environments. This kind of environments generates significant challenges related to the tasks of sensing, control and deliberation. In particular, it is critical to design control algorithms that determine an appropriate action to take based on the current state of the world. But the process to choose these actions could be a difficult task depending of the information known about the environment or the time used to make decisions. A solution could be to get some inspiration from human reasoning processes when solving problems from the real world.

Within the field of **Artificial Intelligence**, **Automated Planning** (AP) (Fikes and Nilsson, 1971a) studies how to choose and execute a set of actions to achieve some goals from an initial state. More specifically, a problem in AP can be defined as a state-transition system, where the states describe the environment and the transitions define the different actions which can be executed in the environment at each problem solving episode. One of these states is defined as the initial state and the system must select which actions allow the system/environment to reach another state where all goals are achieved.

Since the first days of AP, different approaches have been designed to solve problems, but it is just in the last fifteen years that the new approaches based on heuristic search and problem decompositions have allowed to solve difficult planning problems. Even with all these advances in the field, solving complex planning problems from the real world with a planner is not still possible in many occasions. When a planner tries to solve some real world problems, often it cannot generate a complete and detailed plan. Classical AP assumes the world to be static and deterministic. However, the real world usually is stochastic and dynamic. Therefore, the execution of some actions might fail, which can prevent the

execution of the rest of the plan. Researches extended the AP paradigm to reason about uncertainty in the perception of the environment and in the action outcomes. This new sub-field, called **planning under uncertainty**, studies how to generate a plan of actions in stochastic environments. The new techniques generated by planning under uncertainty use a probabilistic model for reasoning about uncertainty, which is often compiled into a Markov Decision Process (MDP). In spite of these new models, solving real world problems with AP is particularly difficult. A way to simplify the complexity of the model of the environment is using **Abstractions**. They have been used often to decrease the complexity of difficult problem task (Newell and Simon, 1972; Korf, 1987). In this thesis, I propose to extend some works on abstractions to the field of planning under uncertainty, by generating plans that provide detailed actions in the first steps of plans. In later steps of the plan, our system will only provide limited details, since the actions that are planned to be executed in the future are very unlikely to be used, given the uncertainty in plan execution. More specifically, in this thesis I propose the definition of different abstractions to decrease the complexity of the environment definition in order to improve planning performance in stochastic and dynamic environments. To design this approach, I have been inspired by the work of Zickler and Veloso (Zickler and Veloso, 2010), where a motion planning technique is presented. It generates a collision-free trajectory from an initial state to a goal state in dynamic environments. They introduced the concept of Variable Level-Of-Detail (VLOD), where the search process focuses on obtaining accurate short-term motion planning steps, while considering the far future with a different level of detail, by selectively ignoring the physical interactions with dynamic objects.

The rest of this document is organized as follows: Chapter 2 summarizes the literature about classical planning, planning under uncertainty, knowledge abstraction in automated planning and planning and execution. Then, Chapter 3 presents the objectives of the thesis. And finally, Chapter 4 describes the methodology to achieve the objectives presented in the previous chapter.

Chapter 2

State of the Art

This chapter is a review of the main works done in the three research topics addressed in this Thesis: Automatic Planning, knowledge abstraction in Automated Planning and planning and execution. Section 2.1 of this chapter introduces the relevant concepts in Automatic Planning based on classical approaches. Section 2.2 presents how abstractions have been used in Automated Planning. Finally, section 2.3 describes some approaches that integrate planning and execution.

2.1 Automated Planning

Automated Planning (AP) is a field of Artificial Intelligence that is based on the **problem space** framework created by Herbert Simon and Allen Newell in 1972 (Newell and Simon, 1972). A problem space is defined by a set of states and a set of actions. Within a problem space, a problem is defined by an initial state and a set of goals. An AP system is based on a set of common features:

- A Conceptual Model. A formal definition of the task to be solved and the structure of the solution
- A Representation Language. This language is used to describe the problem solving task and the environment
- An Algorithm. The technique used to solve the task

At start some assumptions were made to simplify problem solving. They are related to the representation of the environment, the actions or the states of the environment.

- Finite world: the environment is represented as a finite set of states

- Static world: the environment only changes when a action is executed
- Determinism: the execution of the same action in the same state always yields the same new state
- Total observability: there is complete knowledge about the state of the environment
- Implicit time: the execution of an action has no duration. Then, the state transitions are instantaneous
- Reachability goals: the objective of the planning task is to find a set of actions that transform a given initial state to another state satisfying the goals

Depending in which of these assumptions are relaxed we can define different types of planning models. In this thesis, I am interested in the use of Automated Planning in realistic environments. This implies that it is not possible to have complete knowledge about the environment and the action outcomes. In more detail:

- Non-determinism: determinism is an unrealistic assumption, because when an action is executed in a real environment, most times predicting the effects of the action in the environment is difficult. We can differentiate three kinds of outcomes: (1) deterministic, (2) disjunctive when different actions outcomes are possible (actions do not follow a probabilistic model); and (3) stochastic when the effects of the actions follow a probabilistic model.
- Environment observability: in several applications, the state of the environment cannot be observed completely. Depending on the amount of information that is known, three levels of observability can be defined: (1) total observability, when all the information about the environment can be captured or sensed; (2) partial observability when the state cannot be fully observed and (3) no observability when no information about the environment can be captured or sensed, except for the initial state.

According to these concepts, different planning models (and, thus, techniques) can be used to generate plans of actions. Table 2.1 shows some AP models that can be defined depending on these dimensions.
According to Table 2.1, classical or deterministic planning is the simplest sub-field of AP. Planning under uncertainty is the sub-field of AP that studies the relaxation

Planning Paradigms			
Observability	Determinism		
	Deterministic	Disjunctive	Stochastic
Total	Deterministic	Deterministic	Probabilistic
Partial		Contingent	Probabilistic-Contingent
None		Conformant	Probabilistic-Conformant

Table 2.1: Automated planning techniques

of the total observability and deterministic world assumptions. This sub-field can be divided into three fields depending on how the different dimensions are defined. Probabilistic planning works in an environment where action outcomes are stochastic but fully observable. Contingent planning works in deterministic environments where the information known about the world is partially observable. And finally, Conformant planning works in environments where no information about the environment can be observed except for the initial state. Next, the deterministic and probabilistic sub-fields will be explained in more detail.

2.1.1 Classical Planning

Classical Planning can be defined as the process of selecting a set of actions, which sequentially executed transform an initial state into another state where the goals are reached. Figure 2.1 shows an example of the Rovers Domain. This domain was designed for the sequential track of the International Planning Competition (IPC) (2002) and it was inspired on the Mars exploration rover missions where an area of the planet is represented as a set of waypoints. Each waypoint can contain samples of rock or soil which can be collected by the robots. Each robot can go from one waypoint to another, and can perform a set of different actions (analyze rock or soil samples, take pictures of a specific waypoint, ...). All the data collected by the robots has to be sent to the lander, which is placed in a specific waypoint. In this case, the rover has complete knowledge about the environment.

Conceptual Model

The conceptual model for a planning task defines the environment, the actions that can be used to change the environment, and how actions are applied in the environment to achieve the goals. In the case of classical planning, the conceptual model is a deterministic, finite and fully observable state-transition model that can be denoted by $\Sigma = (S, A, \Upsilon, C)$, where:

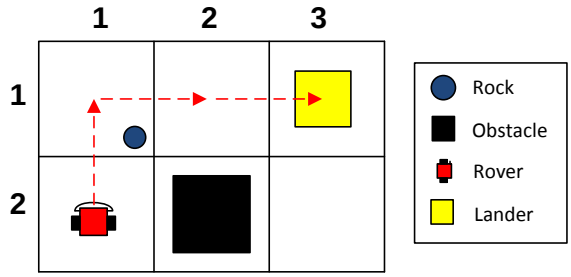

Figure 2.1: An example of a six waypoints problem on the Rovers Domain.

- S is a finite set of states.
- A is a finite set of actions that can be executed in the environment.
- Υ is a transition function $\Upsilon(s, a, s')$, where $\Upsilon : S \times A \rightarrow S$. This function defines the new state s' generated by applying action $a \in A$ in the state $s \in S$
- C is a cost function $C(a)$, that defines the cost of applying action $a \in A$. In case we are only interested on number of actions in a plan, the cost of each action is 1.

According to this model, a classical planning problem can be defined as a tuple $\Pi = (\Sigma, s_0, G)$, where $s_o \in S$ is the initial state and $G \subseteq S$ is the set of goal states. The solution of a classical planning problem is a sequence of actions $(a_0, a_1, \ldots, a_{n-1})$ $\forall a_i \in A$ which sequentially executed transforms the initial state s_0 into a goal state $s_n \in G$.
Figure 2.2 shows the structure of the automata that represents the conceptual model (action/state) for the problem showed on Figure 2.1.

Representation Language

A planning representation language is a notation for the syntax and the semantic of planning tasks. The typical planning descriptions are based on **first-order logic** where each atom of information is defined using a **predicate**. For instance, given two objetcs a *table* and a *glass* and the predicate *on(table, glass)*, this predicate represents that there is a glass on a table. Different languages have been defined in Automated Planning to represent the conceptual language described previously. The most extended language is the Planning Domain Definition Language (PDDL) (Fox and Long, 2003) developed as the planning input language of

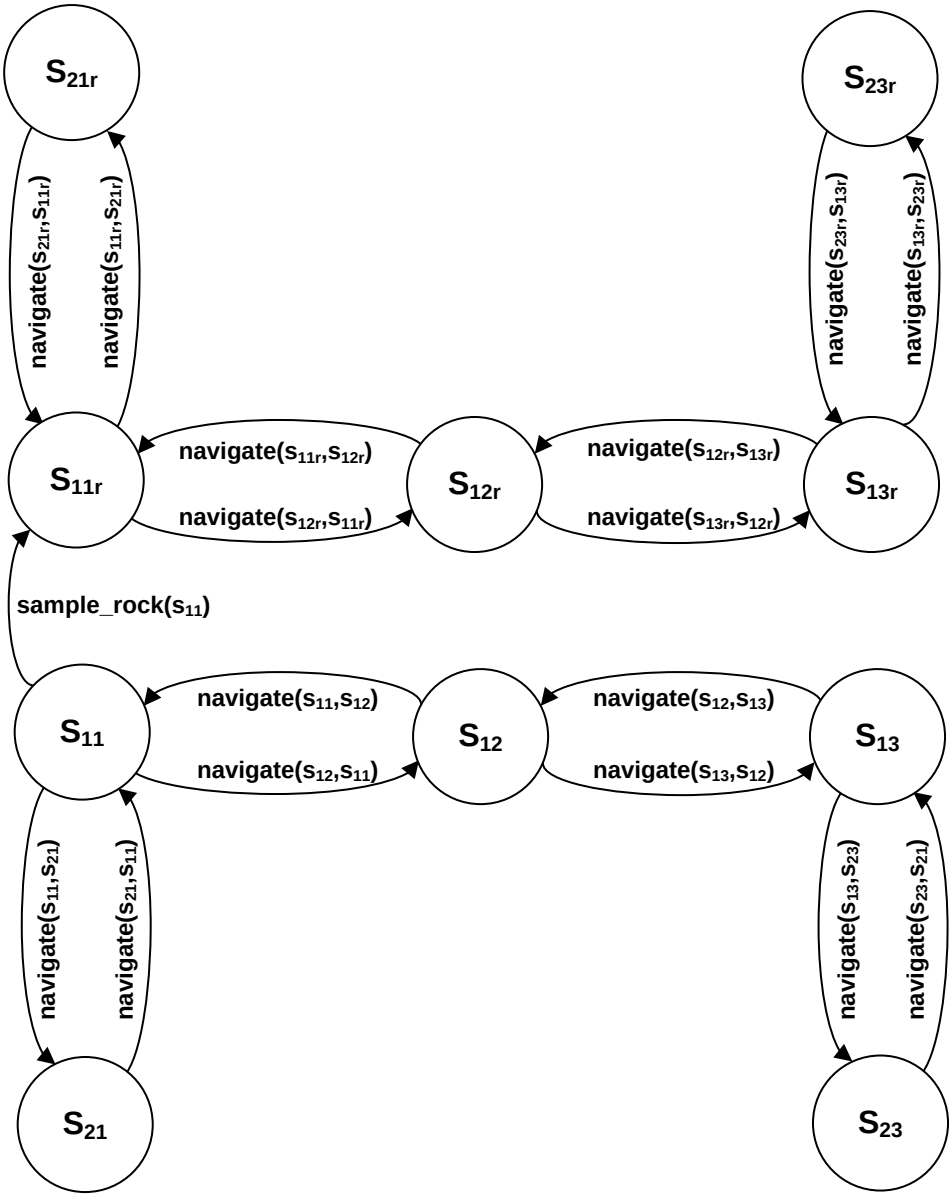

Figure 2.2: Conceptual model corresponding to the robot navigation problem defined in Figure 2.1.

the first IPC (Mcdermott, 2000) in order to define a standard language to allow comparatives among the different planners. PDDL is a very powerful language that has evolved to PDDL3.1 version to cover the representation needs of the AP community. A PDDL planning problem definition is composed of two parts:

1. The domain definition: contains an enumeration of the types, the constants, the static facts, the predicates, the functions and the actions. Each action is represented as a triple $a = (Pre, Add, Del)$, where:

   - Pre defines the preconditions of the action. $Pre(a) \subseteq L$ is a set of literals that have to be true when the action is applied.
   - Add defines the information added by the action. $Add(a) \subseteq L$ is a set of literals that becomes true when the action is applied.
   - Del defines the information deleted by the action. $Del(a) \subseteq L$ is a set of literals that becomes false when the action is applied.

```
(:action navigate
:parameters (?x - rover ?y - waypoint ?z - waypoint)
:precondition (and
                        (can_traverse ?x ?y ?z)
                        (available ?x)
                        (at ?x ?y)
                        (visible ?y ?z)
                        (>= (energy ?x) 8)
                )
:effect (and
                (decrease (energy ?x) 8)
                (not (at ?x ?y))
                (at ?x ?z)
        )
)
```

Figure 2.3: Example of an action in PDDL.

Figure 2.3 shows an example of a PDDL action. In this case, the action corresponds to the action navigate from the Rover domain. In spite of all functionalities of the last version of PDDL, PDDL3.2, most existing planners do not support it. In fact, most planners only support the STRIPS subset that approximately corresponds to what was supported by the STanford Research Institute Problem Solver in 1972 (Fikes and Nilsson, 1971b) besides typing (definition of types) and the use of the equality predicate.

2. The problem definition: contains an enumeration of the objects and the instantiated predicates that define the initial state and the goal of a specific problem in the domain. Both descriptions have to be grounded; all predicate arguments should be objects or constant names rather than variables.

Using this representation, a planner can generate a plan Π of applicable actions $\Pi = (a_0, \ldots, a_{n-1})$ which executed sequentially in the environment transforms the initial state I to a goal state where the literals defined in G are true. Formally, an applicable action a in a state s_i is any action $a \in A$ such that $Pre(a) \subseteq s_i$. The resulting state of applying an action a in a state s_i is a state $s_{i+1} = (s_i / Del(a)) \cup Add(a)$. This implies that the literals that are not defined in the effects of action a remain unchanged in state s_{i+1} when action a is executed. This is known as the **STRIPS assumption** and it means that only actions can change the environment.

Algorithm

There are two main approaches to solve classical planning tasks: (1) search for a solution plan or (2) compiling the classical planning problem into another problem solving task (as SAT (Kautz and Selman, 1996)) and solve it using other kinds of problem solvers. The first approach and most extended one, consists of exploring an implicit graph, searching for a path to a goal state from the initial state. When a problem is solved with a search algorithm three elements have to be considered:

- The search space: is an implicit graph composed of nodes and arcs. Each node corresponds to a state of the environment and each arc corresponds to a state transition resulting from an action execution.

- The search direction: defines the way used to search. If the search starts at the initial state and goes towards the goal state, it is called forward search, but if the search starts at the goal state and goes towards the initial state, it is called backward search. A bidirectional search can also be implemented.

- The search algorithm: defines the method used to explore the Search Space. The first planners, like STRIPS, implemented a depth-first search algorithm. In the following recent years, heuristics have been introduced to guide or prune the search towards the goal (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001). Heuristics are usually based on the definition of an evaluation function, $f(n)$, that scores all states in the search tree according to how close they are to a goal node. The performance of the search process is determined

by the accuracy of the heuristic function guiding it. Heuristics are usually based on a relaxed version of the problem, which is simpler that the original one. But, in the last years, researchers are analyzing other ways to define heuristic functions, like abstractions. In Automated Planning, some of the most common relaxations or abstractions used are:

1. Ignoring the actions delete lists (Hoffmann and Nebel, 2001).
2. Counting some unachieved literals (as landmarks or the initial goals) (Richter and Westphal, 2010).
3. Generating abstractions about the structure of the domain (Merge and Shrink) (Helmert et al., 2007).

The heuristics used in AP are commonly non-admissible. This means that the heuristic function h can overestimate the value of at least one node n in the search space: $\exists\, n\ h(n) > h^*(n)$.

2.1.2 Probabilistic Planning

Probabilistic Planning aims to find a plan that transforms the initial state into a goal state in a stochastic and fully or partially observable environment. The most common way of solving this type of task (Bonet and Geffner, 2005) represents the planning process as an optimization problem using a Markov Decision Process (MDP) and partially observable MDPs (POMDPs) (Cassandra et al., 1994). It based on the following ideas:

- A planning domain is modelled as a stochastic system, where the action outcomes are modelled as a probability distribution function.
- Goals are represented as an utility function, numeric function or a set of goals.
- Solutions are represented as policies that specify the action to execute in each state.

Conceptual Model

The conceptual model for a Probabilistic Planning task is a stochastic, finite and fully observable state-transition model, where each transition has associated a probability distribution and can be denoted by $\Sigma = (S, A, P, \Upsilon, C)$, where:

- S is a finite set of states, which is composed of all states that can be reached.
- A is a set of actions whose effects follow a probabilitic model.
- $P(s'|s,a)$, is the probability that the action $a \in A$ executed in state $s \in S$ results in a state $s' \in S$. This means that for each state $s \in S$, if there exists an action $a \in A$ and a state $s' \in S$ such that $P(s'|s,a) \neq 0$, then $\sum_i P(s_i|s,a) = 1$.
- Υ is a transition function $\Upsilon(s,a,s')$, where $\Upsilon : S \times A \rightarrow 2^S$. This function defines the new state generated by applying the action $a \in A$ in state $s \in S$.
- C is a cost function $C(a)$, which defines the cost of applying action $a \in A$.

According to this model a probabilistic planning problem can be defined as a tuple $\Pi = (\Sigma, s_0, G)$. The solution of this problem is a sequence of actions $(a_0, a_1, \ldots, a_{n-1}) \forall\ a_i \in A$ or a policy $(s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}) \forall\ a_i \in A\ \forall\ s_i \in S$ which sequentially executed transforms the initial state s_0 into a goal state $s_n \in G$. The quality of a solution to a probabilistic planning problem depends on two factors: the cost of the actions of the plan Π calculated as $\sum_{i=1}^{n} C(a_i)$ and the probability of achieving the goals calculated as $\prod_{i=0}^{n} P(s_i|s,a)$.
Figure 2.4 shows the conceptual model corresponding to the robot navigation problem showed on Figure 2.1. In this case, the rover's actuators can fail when moving right or left. Specifically, in this environment when the rover moves left, it succeeds with probability 0.5 and does nothing with probability 0.5. And when moving right it succeeds with probability 0.8, doing nothing with probability 0.2.

Representation Language

The Probabilistic Planning Domain Definition Language (PPDDL) (Younes and Littman, 2004) was the standard representation language that allowed to describe Automated Planning problems in stochastic domains in the first IPC with a non-deterministic track (Younes et al., 2005). This language is an extension of PDDL that support actions with probabilistic effects and probabilistic literals on the initial state. Each set of literals is defined with a probability p. Equation 2.1 shows the structure of probabilistic effects on PPDDL, where $\forall e_i\ \exists\ p_i$.

$$(probabilistic\ p_1\ e_1\ p_2\ e_2\ \ldots\ p_{n-1}\ e_{n-1}\ p_n\ e_n) \tag{2.1}$$

Obviously, $p_i >= 0$ and $\sum_{i=0}^{n} p_i \leq 1$. If the sum of the probabilities is lower than 1, it is assumed that there is a probability $P = 1 - \sum_{i=0}^{n} p_i$ that the action

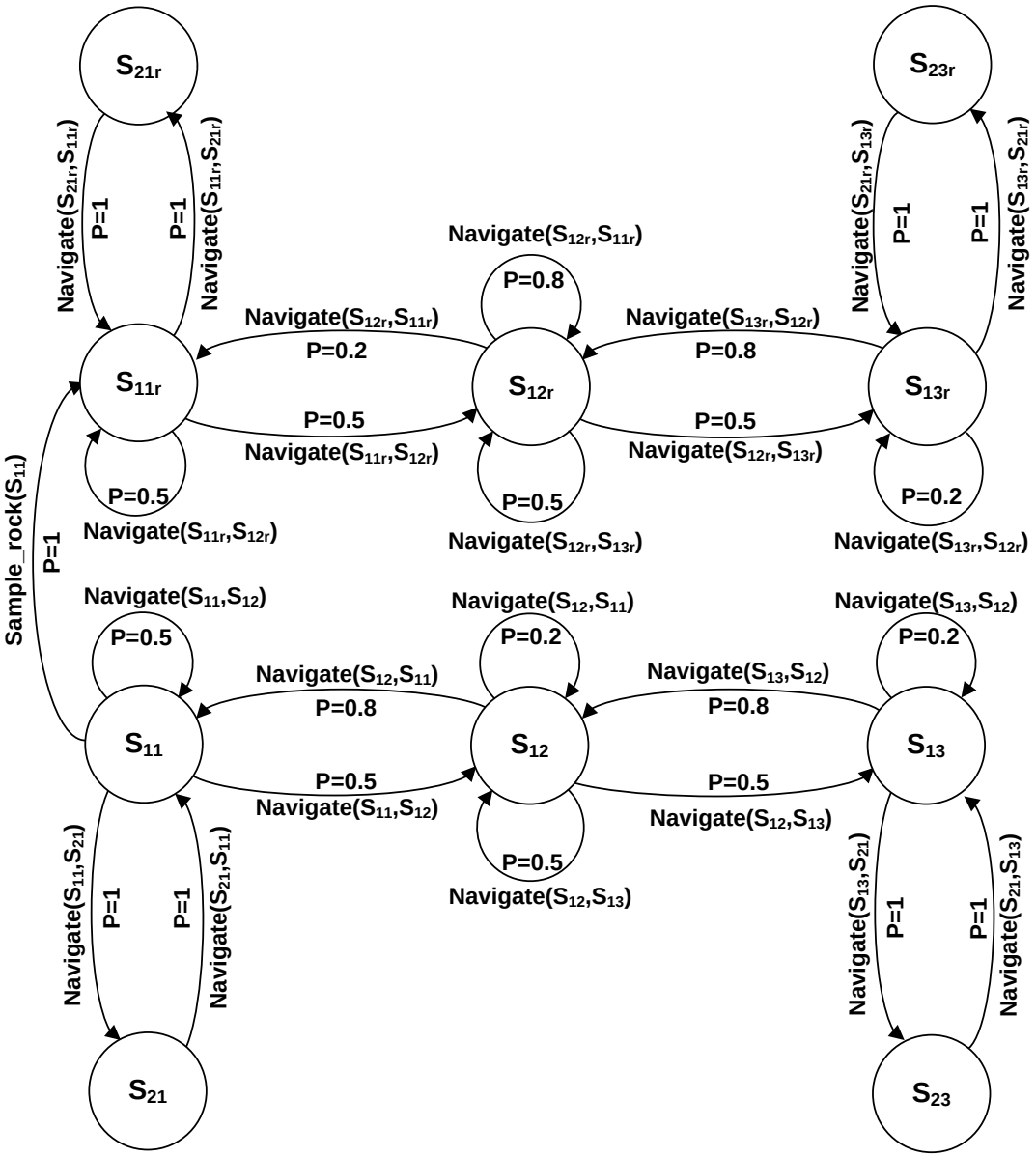

Figure 2.4: Conceptual model corresponding to the robot navigation in a probabilistic environment.

does not generate any outcome. Figure 2.5 shows an example of a probabilistic action in the Rovers Domain, In this action the rover *?rover* will move to waypoint *?to_waypoint* from waypoint *?from_waypoint* with 0.8 probability and it will not move with a 0.2 probability.

```
(:action navigate
      :parameters (?rover ?from-waypoint ?to-waypoint)
      :precondition
         (and
                  (at ?rover ?from-waypoint)
         )
      :effect
         (and
                  (probabilistic 0.8
                           (at ?rover ?to-waypoint)
                           (not (at ?rover ?from-waypoint))
                  )
         )
)
```

Figure 2.5: Example of a probabilistic action from Rovers domain.

In 2011, RDDL (Relational Dynamic influence Diagram Language) (Sanner, 2011) was presented and used as the official language of the uncertainty track of the 7th IPC (Coles et al., 2012). Conceptually, this new language is based on PPDDL1.0 and PDDL3.0, but practically it is a completely different language both syntactically and semantically. The introduction of partial observability is one of the most important changes in RDDL compared to PPDDL1.0. It allows for the efficient description of Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) by representing everything (state-fluents, observations, actions) with variables.

Algorithms

The algorithms used to generate a plan with Probabilistic Planning try to find policies (mappings between world states and the preferred action to be executed to achieve the goals) that optimize a utility function, which gives preference to the different states and transitions of the MDP. There are three main approaches to manage probabilistic planning problems:

- Policy iteration: The basic idea of this approach is to start with a randomly selected initial policy and refines it repeatedly. Commonly, the algorithm

alternates between two phases: (1) an evaluation phase, in which the cost of the actual policy is calculated and (2) a policy refinement phase, in which the actual policy is refined to a new policy with a smaller expected cost.

- Value iteration: The basic idea of this approach is to define a randomly selected cost for each state $c_n(s_n)$ on the MDP and refine the value of each state finding an action that minimizes the expected cost. This kind of algorithms is composed of two phases: (1) a value determination phase, in which the expected cost of each state is calculated and (2) a value refinement phase, in which the algorithm finds an action that minimizes the cost of a state and stores it in the policy. This technique can be used with heuristics to improve its performance (Kolobov et al., 2010).

- Heuristic search: Heuristic search algorithms are a commonly used solution in classical planning which have been used to solve MDPs. Some examples are: the LAO* algorithm (Hansen and Zilberstein, 2001), which is a generalization of the A* algorithm for MDPs; the Learning Depth-First Search (LDFS) algorithm (Bonet and Geffner, 2006), which is a generalization of the IDA* for MDPs; the Stochastic Enforced Hill Climbing (SEHC) algorithm (Wu et al., 2008), which extends the heuristic search algorithm EHC for classical planning to probabilistic planning, or the planner mGPT (Bonet and Geffner, 2005) that solves MDPs using different heuristic-search algorithms.

2.2 Abstractions

An abstraction in AP can be defined as a function that transforms a problem space into another (simpler) one, where some details are ignored. Usually, abstractions help to reduce the problem complexity, at the cost of soundness.

2.2.1 Abstractions over the search space

There have been already many approaches that generate abstractions for Automated Planning. These approaches are related with the structure of the search space decreasing its size to reduce the problem complexity. The first work that used abstractions in Automated Planning was AbSTRIPS (Sacerdoti, 1974) that extended the work of Newell and Simon on GPS (Simon and Newell, 1969). AbSTRIPS builds abstract problem spaces by assigning criticalities to the preconditions of the operators. The criticality of a precondition is a natural number

that characterizes the importance of the precondition in the planning domain. The predicates are grouped in abstract spaces depending of the value of their criticalities. When the criticalities have been defined, the search process starts generating an abstract plan achieving the most important variables. Next, the search process gradually refines the abstract plan by adding operators that achieve values of less important variables. AbSTRIPS can delete some variables which have been achieved in previous levels degrading the performance of the planner. Alpine (Knoblock, 1991) automatically generates abstraction hierarchies using the preconditions and effects of the operators. This system solved some of the problems presented by AbSTRIPS related with the process that assigns the criticalities and automatically built the abstract spaces. Both systems use abstraction in a similar way although they assign criticalities in different ways.

2.2.2 Abstractions over the heuristic function

Heuristics are a way of ranking a set of nodes in order of define their quality and choose what is the best successor to explore the search space. They are modelled as a function h that returns a number, for each node of the search space. Commonly, heuristic functions are used to define the distance from a node n to a goal node. In the last years, these functions have been designed using abstractions over the problem:

- The FF heuristic (Hoffmann and Nebel, 2001), h_{FF}, is a domain independent heuristic function derived as the cost of the plan of a relaxed problem. The planning problem relaxation consists of ignoring the delete list of all actions and extracting a solution using a Graphplan-style algorithm (Blum and Furst, 1995). The number of actions in the relaxed solution is used as a goal distance estimate. The relaxation can be considered as an abstraction. The process of creating a graph of the search space where delete lists are ignored for each action is a simplification of the original problem decreasing the complexity of the plan generation process.

- Pattern DataBases (PDB) (Edelkamp, 2001) are a set of precomputed tables of the exact cost of solving various sub-problems of an existing problem. Each sub-problem is an abstraction of the original problem. The cost of solving the abstract sub-problem is a lower bound on the corresponding cost in the state space of the original problem.

- The Merge-and-Shrink heuristic (M&S) (Helmert et al., 2007) builds an abstraction space from atomic abstractions that are directly associated with

the variables of the planning problem. The definition of the variables is based on the SAS+ planning model (Bäckström and Nebel, 1995), where each variable is defined as a multi-valued state variable that is composed of a set of literals. The abstract state space in this heuristic is built incrementally, starting with a set of atomic abstractions associated with each individual SAS+ variable and merging two abstractions (replacing them with their synchronized product) and shrinking them (aggregating pairs of states in one) at each step. This heuristic uses two different abstractions: (1) defining the predicates as SAS+ variables and (2) generating abstract spaces to apply the process of merge and shrink.

2.3 Planning and execution

The execution of a plan of actions that theoretically solves a problem, can fail for many different reasons, as for instance: environment information that was not captured in the initial representation; execution may yield unexpected states. Thus, when integrating planning an execution, most people resort to architectures that allow planning, monitoring, execution and replanning. The simplest architecture that can be defined to interleave planning and execution, shown in Figure 2.6, is composed of three elements:

- A planner module, that generates a plan for a planning task. Commonly, the planner uses a deterministic action model that it is simple and incomplete, given that it does not consider any contingency.

- An execution module, that is a sensor-actuator system. This module must recieve the information from the environment and executes the actions in the plan.

- A monitoring module, that sends actions to the execution module and observes the result of each one, analyzing if an unexpected state has been observed and decides whether execution should proceed, or it should replan. Different strategies for monitoring can be used, such as: plan monitoring (PM), where a sub-plan is considered executable when all the preconditions of the actions that are not established by actions of the sub-plan are currently holding in the environment. If the sub-plan cannot be executed, the monitoring module has to request a new plan from the execution module; or action monitoring (AM), where an action is considered executable if the

preconditions are true when it is executed. But if the preconditions of the actions are not true, the plan cannot be executed and a new plan has to be generated from the current state.

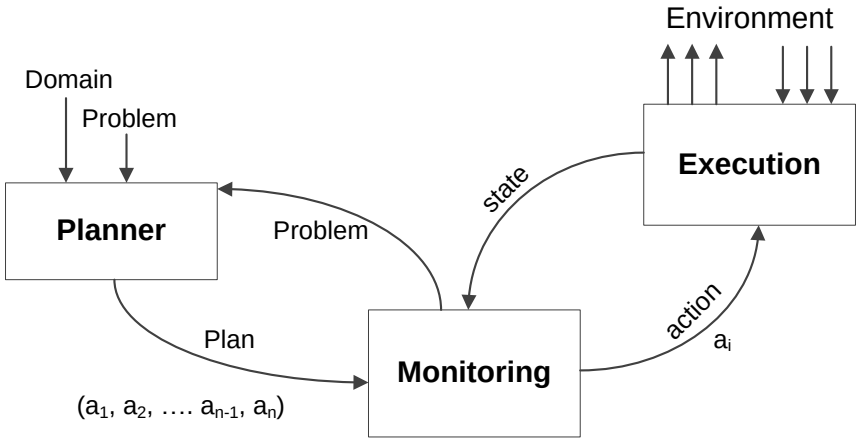

Figure 2.6: Example of a re-planning process.

Besides, there are different techniques that can be applied to generate a plan depending of the known information about the environment. If we have information about the dynamics of the environment (accurate model of the actuators of a robot, failure probabilities of actions, accuracy models of sensors, ...), we can define a domain model with probabilistic information (such as in PPDDL or RDDL). In this case, we can generate a conditional plan (Peot and Smith, 1992) that takes into account all possible states, or we can generate a policy by solving an MDP as shown by LAO (Hansen and Zilberstein, 2001), or LRTDP (Bonet and Geffner, 2003). But, usually, the dynamics of the environment are not known, so we have two alternatives. First, we can learn the dynamics, but usually the learning effort is impractical except for small problems (Zettlemoyer et al., 2005). The second solution, shown on Figure 2.6 and the most common one, consists of using a deterministic domain model adapting to the changes when an unexpected state is detected. There are two alternatives to react when an unexpected state makes the execution of the rest of the plan unfeasible:

- A **re-planning** process, the system generates a new plan. This process is repeated until the system reaches the problem goals. Therefore, at each planning (re-planning) step, including the initial one, the system is devoting a huge computational effort on computing a valid plan (applicable plan that achieves the goals), when most of it will not be applied (Yoon et al., 2007).

- A **repairing** process, the system adapts the existing plan to the new state of the environment and/or the new planning goals (Gerevini and Serina, 2000; Koenig et al., 2002; Fox et al., 2006; Borrajo and Veloso, 2012). But repairing techniques are not useful when the environment is highly dynamic or complex, where the benefits of this technique decreases.

There are already many approaches that interleave planning and execution. Some of these architectures are:

- PELA (Planning, Execution and Learning Architecture) (Jiménez et al., 2008) is an architecture that integrates planning, execution and learning techniques. This architecture generates a plan using a planner with a deterministic model; executes the plan storing the result as success, failure or dead-end; and learning the patterns for predicting these outcomes.

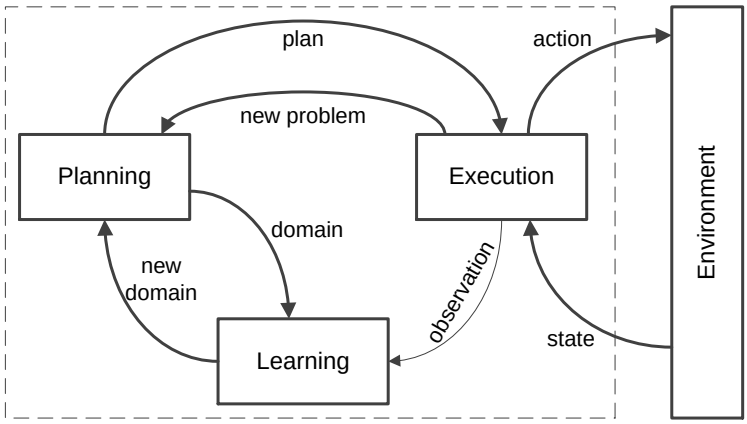

Figure 2.7: PELA architecture.

Figure 2.7 shows the structure of the architecture along with the integration of the modules. As it can seen, PELA has three components. The planning and learning components can be exchanged for others that provide the same functionality with the same inputs and outputs.

- T-REX (Teleo-Reactive Executive) (McGann et al., 2008a) is an architecture designed first for the control of the Mars rovers, and later used for planning and execution applied to the control of autonomous underwater vehicles in real oceanographic scientist missions. This architecture is composed for different modules or Tele-Reactors (McGann et al., 2008b) organized in a hierarchical structure, where each reactor uses a temporal constraint satisfaction

based planner. Tele-reactors receive goals to generate their behaviour, send goals to lower level reactors, and receive observations from lower level reactors. T-REX uses a specific language (NDDL) to describe domain and problems.

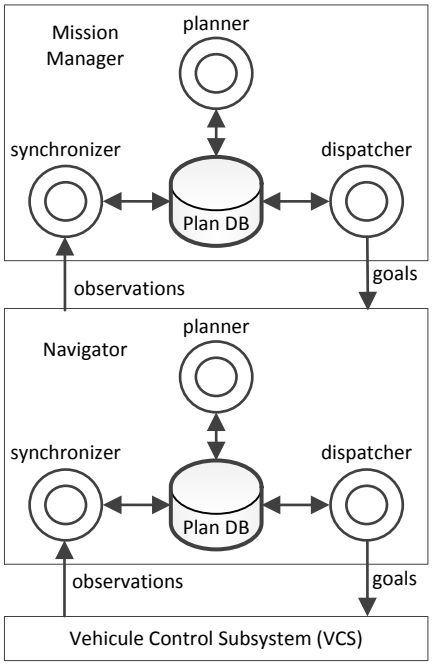

Figure 2.8: T-REX architecture.

Figure 2.7 shows an example of the structure of the T-REX architecture. As it can seen, in this case the architecture is composed of three modules: two reactors (mission manager, navigator) and a function layer (vehicle control subsystem). In this case, VCS works as a functional layer that connects with the autonomous underwater vehicle. This architecture divides the planning task among the different reactors.

- PELEA (Planning, Execution and LEarning Architecture) (Guzmán et al., 2012) is an architecture that includes components that allow to integrate planning, execution, monitoring, re-planning and learning techniques in a dynamic way. PELEA provides two main types of reasoning: high-level (mostly deliberative) and low-level (mostly reactive). This is common to most robotics applications which reflects the division between the deliberative components and the reactive components. However, in this architecture, these are two planning levels, which offer two main advantages: both levels

can be easily adapted to the requirements of the automated systems; and the division in two levels allows for the automatic systems recovering from failed executions at either level. For instance, if a reactive failure is detected, it can be solved in the low-level layer and generating a new high level plan is not needed.

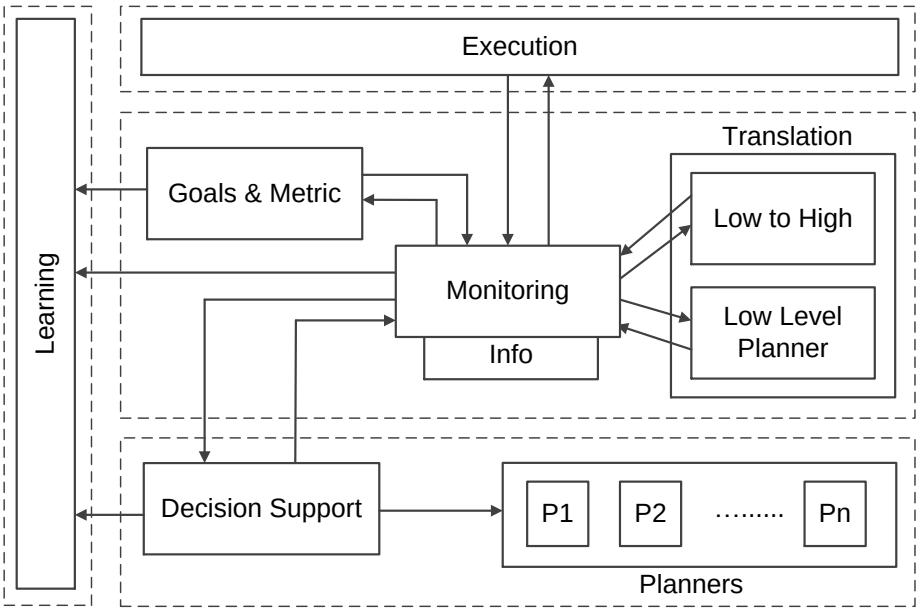

Figure 2.9: PELEA architecture.

Figure 2.9 shows the structure of the PELEA architecture along with the integration of the modules. As it can be seen, PELEA is composed of eight components that exchange a set of Knowledge Items (in XML) during the reasoning and execution steps. The knowledge used by the architecture is composed of the information about the environment (Domain representation, Problem/State representation) in different levels of detail (High and Low).

2.4 Discussion

Automated Planning offers a good way to provide deliberation by generating plans to interact with the real world. But when interleaving planning and execution the deliberation time must be short, because most control systems need a quick reasoning cycle (Simon, 1955). Besides, the planning task must adapt to the contingencies of the environment (new information can be discovered, failures

on the execution of the actions, etc). For these reasons, classical planners scale better (FFReplan (Yoon et al., 2007) achieved the best overall performance at the probabilistic competition of IPC-2004 and IPC-2006) than the planners that explicitly manage uncertainty. It part, it had to do with the fact that the domains used in most IPCs in the non-deterministic track did not include execution dead-ends. But, in spite of this, when classical planners are used to solve problems from the real world they do not scale really well. Both paradigms, classical planning and planning under uncertainty offer advantages and disadvantages to work in real world environments. In both cases, the planner devotes a huge computational effort on computing a valid plan (applicable plan that achieves the goals), when most of it will not be applied due to unexpected outcomes of actions execution. With this thesis, I propose to work on the problem of finding plans fast that are k-bounded sound. It is based on two ideas: some effort is devoted to compute a valid head of the plan of a specific length k; and the rest of the plan is checked for potential reachability by relaxing the complexity of actions and decreasing domain details through abstractions.

Chapter 3

Objectives

Current Automated Planning models (classical planning, planing under uncertainty) based on heuristic search or dynamic programming, only obtain robust plans when they have complete and accurate action models. But unfortunately obtaining an accurate action model from the real world is not always possible or the cost to get it is huge. Besides, if the information used to represent the environment is huge, the time required to generate a plan could be prohibitively large. The main objective of this thesis consists in proving the initial hypothesis:

> It is possible to solve a planning problem using a deterministic action model by Automated Planning by generating k-bounded plans, using abstractions which are built by removing some predicates that represent future information of the environment.

This hypothesis is based on two ideas: it is not always possible to collect all the information about the dynamics of a real world environment or if it is possible to get it, the amount of information that describes the environment can not be managed by current planners; and in dynamic and stochastic environments it is not useful to spend a long time generating a detailed long plan, when most of it will not be applied by the dynamic behaviour of the environment (changes in the information known, unexpected states, new information, etc). The main objective has been divided into several sub-objectives, which are described in detail next:

- **Analyzing the existing literature related with this thesis** (Objective 1): perform an exhaustive review of the existing literature in classical planning, planning under uncertainty, abstractions on AP and planning and execution. It requires a deep study about the advantages and limitations of the existing paradigms and how they can be improved to solve problems similar in the real world.

- **Decreasing the computational effort of the planning task** (Objective 2): solving problems in dynamic and stochastic environments is challenging in Automated Planning. I propose a new planning approach to decrease the time of the planning task that applies abstractions over the information known about the environment. The abstractions will be created over information about the future, which could change during the planning and execution processes.

- **Designing an automatic way to generate abstractions** (Objective 3): it is very important to identify **what** information can be used to generate the abstractions and **why**. This objective consists on designing a technique that generates abstractions automatically. This technique must analyze the information used to represent the environment and generate a set of abstractions, which can be used by the technique designed to solve the objective 2. The rules used to select the information to be abstracted must be based on the importance of the information for the planning process.

- **Building a set of test benchmarks** (Objective 4): The currently test benchmarks for Automated Planning have been designed to analyze the power of planners to find a solution, but these solutions are not usually executed in an environment. For this reason, it will generate a set of test benchmarks and define a way to evaluate the features of the different techniques developed in this Thesis.

- **Deploying the techniques developed in a real environment** (Objective 5): The planning techniques are usually tested in simulators, which are really far away from the real world. The last objective of this thesis consists on deploying the different techniques in real world environments or over simulators whose dynamics are very similar to the real world. More specifically:

    1. Controlling the interaction with the world of the humanoid robot Nao. Interaction with the real world is the most important task for a robot like this, that can manipulate objects and use different sensors to observe the environment.
    2. Selecting some environments with similar features to real world tasks that allow deploying planning techniques. For instance, a multi-agent simulator or a Real Time Strategy game (RTS).

Chapter 4

Methodology

This chapter describes the methodology that will be used to carry out this thesis. Section 4.1 presents the description of the different steps of this thesis. Section 4.2 shows a detailed description of the evaluation that will be done over the different techniques developed. Finally Section 4.3 shows the different publications generated by the activities of this thesis.

4.1 Work plan

The following schedule is set up with the aim of guiding the work towards the achievement of the Thesis objectives. Activities A1.2, A1.3 and A2.1 of the work plan are already finished. Figure 4.1 shows the schedule of the different tasks of this thesis.

- Task 1 (Task1): Exhaustive review of the existing literature about the different elements related with the thesis:

    1. Activity 1.1 (A1.1): A deep study about the features and the limitations of the existing planning paradigms (classical planning, planning under uncertainty).
    2. Activity 1.2 (A1.2): A deep review about abstractions and how these have been used in Automated Planning. Analysing the structure of the abstractions, the way used to generate them and how these are applied in the different Planning Paradigms.
    3. Activity 1.3 (A1.3): A deep analysis about the different methods designed by interleaving Planning and Execution.

Id.	Activity	Start Day	End Day
1	Start Thesis	01/09/2011	01/09/2011
2	**Task1**	**02/09/2011**	**21/03/2014**
3	A1.1	02/09/2011	21/03/2014
4	A1.2	01/11/2011	31/01/2014
5	A1.3	01/11/2011	30/05/2013
6	**Task2**	**11/06/2012**	**13/12/2013**
7	A2.1	11/06/2012	30/09/2013
8	A2.2	01/11/2012	13/12/2013
9	**Task3**	**20/08/2012**	**08/09/2014**
10	Evaluation 1	20/08/2012	18/11/2013
11	Evaluation 2	01/02/2013	08/09/2014
12	Evaluation 3	03/03/2014	08/09/2014
13	**Task4**	09/06/2014	06/03/2015
14	**Task5**	09/03/2015	03/07/2015
15	End Thesis	06/07/2015	06/07/2015

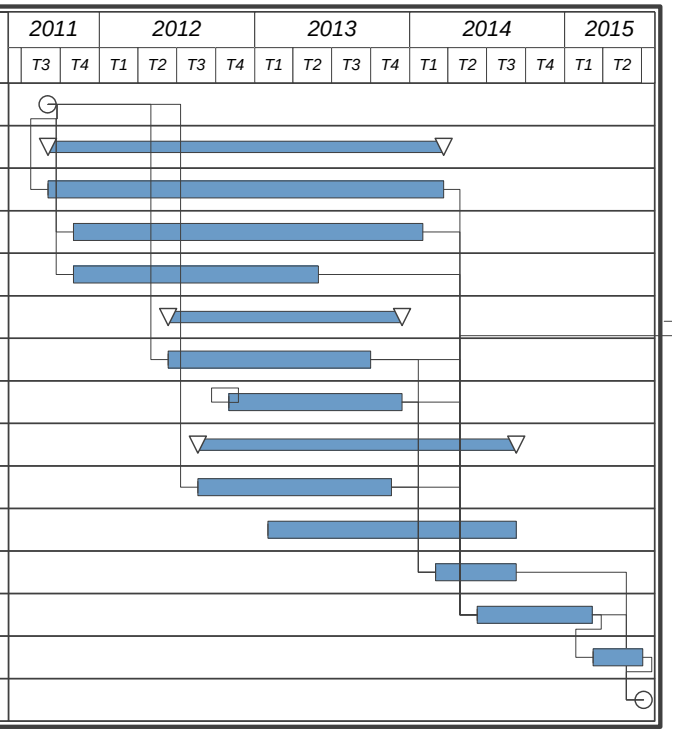

Figure 4.1: Thesis Schedule.

- Task 2 (Task2): Implementing the different techniques proposed in this thesis:

  1. Activity 2.1 (A.2.1) Generating an Automated Planner that uses abstractions about the information of the environment.
  2. Activity 2.2 (A.2.2) Developing a method that automatically generates abstractions.

- Task 3 (Task3): Analyzing the performance of the techniques. To analyze the performance of the new techniques, I will need to deploy the planner developed using a planing-execution system. In order to carry out this task, I will use a new version of the PELEA architecture developed for this thesis.

- Task 4 (Task4): Evaluating the performance of the new techniques with the state-of-the-art non-deterministic planners using different configurations and test benchmarks. The evaluation process will be described in detail next.

- Task 5 (Task5): Writing the Thesis document. It will describe the different techniques developed and the result obtained in the evaluation.

4.2 Evaluation

To evaluate the performance of the techniques developed in this thesis I will be using the International Planning Competition (Mcdermott, 2000), which offers a framework to evaluate the new developed planning systems. This framework provides a standard representation language (PDDL for deterministic planners and PPDDL for non-deterministic planners), test benchmarks, problem generators and metrics to compare the techniques' performance. I propose an evaluation of the thesis objectives based on the resources provided by this competition and other designed specifically for this thesis. The evaluation process is divided in three steps:

1. Test the performance of the proposed abstraction mechanism (Objective 2) in simulated environments (Objective 4). To evaluate the performance of the abstract planning technique I will use MDPSim (MDP Simulator) which is the software provided by the probabilistic track of IPC-2004 and IPC-2006. This simulator allows to emulate stochastic worlds where actions may have diverse outcomes. The current state of the simulator can be totally observed at any time, and it is updated only when a new action is executed. Figure 4.2 shows an overview of the architecture proposed to test the performance of the new planning system. Deploying these experiments implies the definition of four input variables:

    - Planner: experiments will be performed with three planners: Metric-FF (Hoffmann, 2003) is a forward search planner; Reactive-FF is a variation of Metric-FF that generates a plan composed by one action. This planner uses a deterministic domain and selects the best next action according to the heuristic used by the planner; and Abstract Metric-FF that implements the abstraction mechanism that will be generated in this thesis.
    - Problems: a set of difficult problems from different IPC domains (Rover, Satellite, DriverLog, Gold-Miner, etc) will be selected to analyze the effects of the technique. Each problem will be executed several times.
    - Predicate: it is used to build the abstractions. In these experiments abstractions will be generated manually analyzing the structure of the domains.
    - Horizon: it defines when the abstraction is applied during the search

process. In these experiments a set of horizons (1, 3, 5, 10, 20, ...) will be selected manually.

The performance of each problem will be measured by the average values of six dimensions:

- The number of times that the planner has solved the given problem
- The time that the planner has spent to generate the first solution
- The total planning time that the planner has spent solving the given problem
- The total planning time that the planner has spent solving and executing the given problem
- The number of actions that the planner needed to solve a given problem (quality of the plan)
- The number of replanning steps that it needed to solve the given problem

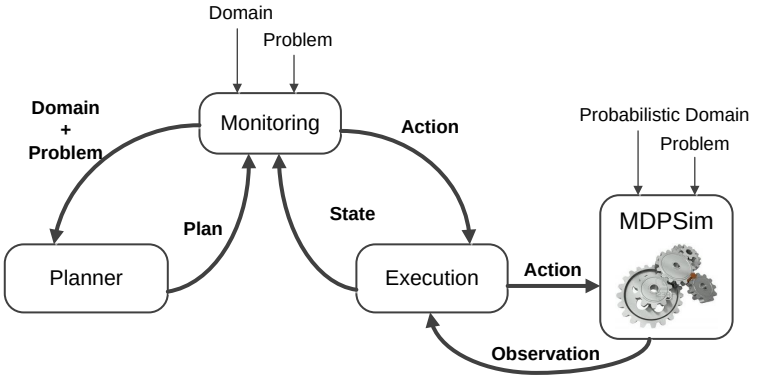

Figure 4.2: PELEA Architecture proposed to test the new planner in a stochastic environment.

2. Compare the performance of the proposed abstraction mechanism (Objective 3) with other techniques (Objective 4). To evaluate the performance of the new abstract planning technique in a similar environment. Deploying experiments implies the definition of three input variables:

   - Planner: experiments will be performed with six planners selected from the planning under uncertainty tracks of the last IPCs:

  - FF-Replan (Yoon et al., 2007) compiles the input probabilistic domain into a deterministic domain generating a plan using the FF planner. If the execution of the plan reaches an unexpected state, FF-Replan replans with the same compilation of the problem to find a plan for this state.
  - RFF (Teichteil-königsbuch et al., 2008) computes an off-line policy combining classical planning and simulation. RFF compiles the probabilistic problem into a deterministic problem with exactly one deterministic action per outcome of a probabilistic action. Then it computes a solution plan with the FF planner. Next, RFF uses Monte-Carlo simulation to estimate the probability of failure of the plan steps.
  - FPG (Buffet and Aberdeen, 2007) uses gradient ascent for direct policy search and factors the parametrized policy by using a function approximation for each action.
  - LPG-Adapt (Fox et al., 2006) computes an initial plan using the LPG planner, but if there is a previous plan the planner reuses it.
  - Abstract-FD is a variation of Fast Downward (Helmert, 2006) that generates a plan using the abstraction mechanisms that will be generated in this thesis selecting abstractions automatically.
  - Reactive-FF is a variation of Metric-FF that generates a plan composed by one action. This planner uses a deterministic domain and selects the best next action according to the heuristic used by the planner.

- Problems: a set of difficult problems from different domains (Rover, Gold-Miner, Wumpus, Localize, Schedule, Drive, Elevators, etc) will be selected to analyze the efects of the technique. Each problem will be executed several times.
- Horizon: it defines when the abstraction is applied during the search process. In these experiments a set of horizons (1, 3, 5, 10, 20, ...) will be selected manually.

The performance of each problem will be measured by the average values of six dimensions:

- The number of times that the planner has solved the given problem
- The time that the planner has spent to generate the first solution

- The total planning time that the planner has spent solving a given problem
- The total planning time that the planner has spent solving end executing the given problem
- The number of actions that the planner has needed to solve the given problem (quality of the plan)
- The number of replanning steps that it has needed to solve the given problem

The same architecture proposed previously 4.2 will be used to execute these benchmarks.

3. Test the performance of the proposed abstraction mechanism in real environments (objective 5). I will evaluate this technique with real robots (Nao, P3DX), where the probability of failures is not known. Figure 4.3 shows the architecture proposed to test the planning system in real world environments. In this case the information used by the architecture is divided in two levels: high level (StateHL, Plan) is the state of the environment in PDDL; and low level (StateLL, ActionLL, Observation) is the state of the environment in the low level representation.

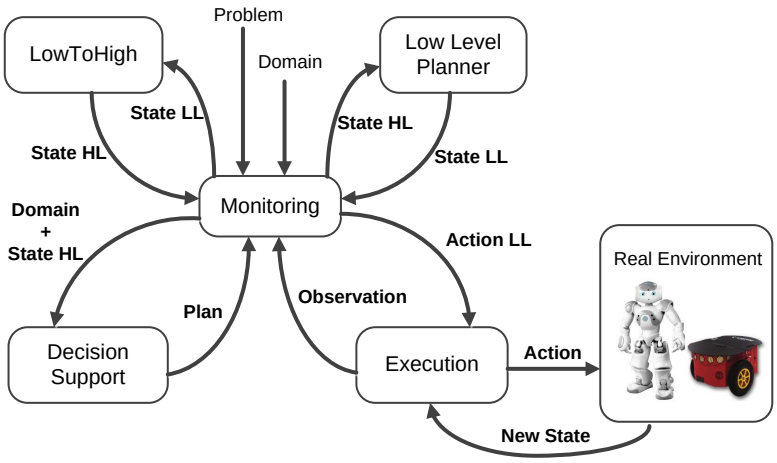

Figure 4.3: PELEA Architecture proposed to test the new planner in a real environment..

4.3 Publication List

Currently, we have published the preliminary results of the thesis in:

- Variable resolution planning through predicate relaxation. Moisés Martínez, Fernando Fernández and Daniel Borrajo. In Proceedings of ICAPS'12 workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx), 5–12, Atibaia (Brazil) 2012.

Bibliography

Bäckström, C. and Nebel, B. (1995). Complexity results for sas+ planning. *Computational Intelligence*, 11(4):625–655.

Blum, A. L. and Furst, M. L. (1995). Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1):1636–1642.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129:5–33.

Bonet, B. and Geffner, H. (2003). Labeled rtdp: Improving the convergence of real-time dynamic programming. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*, pages 12–21, Trento, Italy.

Bonet, B. and Geffner, H. (2005). mgpt: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research*, 24:933–944.

Bonet, B. and Geffner, H. (2006). Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to mdps. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, pages 142–151, Cumbria, UK.

Borrajo, D. and Veloso, M. (2012). Probabilistically reusing plans in deterministic planning. In *Proceedings of ICAPS'12 workshop on Heuristics and Search for Domain-Independent Planning*, Atibaia, Brazil. AAAI Press.

Buffet, O. and Aberdeen, D. (2007). Ff + fpg: Guiding a policy-gradient planner. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 42–48, Providence, Rhode Island, USA.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 1023–1028, Seattle, WA, USA.

Coles, A. J., Coles, A., Olaya, A. G., Jiménez, S., López, C. L., Sanner, S., and Yoon, S. (2012). A survey of the seventh international planning competition. *AI Magazine*, 33(1).

Edelkamp, S. (2001). Planning with pattern databases. In *Proceeding of the sixth European Conference on Planning*, pages 13–24.

Fikes, R. and Nilsson, N. J. (1971a). Strips: A new approach to the application of theorem proving to problem solving. *Journal of Artifitial Intelligence*, 2(3-4):189–208.

Fikes, R. E. and Nilsson, N. J. (1971b). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Fox, M., Gerevini, A., Long, D., and Serina, I. (2006). Plan stability: Replanning versus plan repair. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, pages 212–221.

Gerevini, A. and Serina, I. (2000). Fast plan adaptation through planning graphs: Local and systematic search techniques. In *AIPS*, pages 112–121.

Guzmán, C., Alcázar, V., Prior, D., Onaindía, E., Borrajo, D., Fdez-Olivares, J., and Quintero, E. (2012). Pelea: a domain-independent architecture for planning, execution and learning. In *Proceedings of ICAPS'12 Scheduling and Planning Applications woRKshop (SPARK)*, pages 38–45, Atibaia (Brazil). AAAI Press.

Hansen, E. A. and Zilberstein, S. (2001). Lao*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Helmert, M., Haslum, P., and Hoffmann, J. (2007). Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 176–183, Providence, Rhode Island, USA.

Hoffmann, J. (2003). The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341.

Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Jiménez, S., Fernández, F., and Borrajo, D. (2008). The pela architecture: Integrating planning and learning to improve execution. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1294–1299, Chicago, Illinois, USA.

Kautz, H. A. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Portland, Oregon, USA.

Knoblock, C. (1991). Characterizing abstraction hierarchies for planning. In *Proceedings of the Ninth National Conference of Artifical Intelligence*, Anaheim, CA.

Koenig, S., Furcy, D., and Bauer, C. (2002). Heuristic search-based replanning. In *Proceedings of Tuelveth International Conference on Artificial Intelligente Planning and Scheduling*, pages 294–301.

Kolobov, A., Mausam, and Weld, D. S. (2010). Classical planning in mdp heuristics: with a little help from generalization. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, pages 97–104, Toronto, Ontario, Canada.

Korf, R. E. (1987). Planning as search: A quantitative approach. *Artificial Intelligence*, 33(1):65–88.

Mcdermott, D. (2000). The 1998 ai planning systems competition. *AI Magazine*, 21:35–55.

McGann, C., Py, F., Rajan, K., Ryan, J. P., and Henthorn, R. (2008a). Adaptive control for autonomous underwater vehicles. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1319–1324, Chicago, Illinois, USA.

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., and McEwen, R. S. (2008b). A deliberative architecture for auv control. In *2008 IEEE International Conference on Robotics and Automation*, pages 1049–1054, Pasadena, California, USA.

Newell, A. and Simon, H. (1972). *Human Problem Solving*. Prentice Hall, Englewood Cliffs, Upper Saddle River, NJ, USA.

Peot, M. A. and Smith, D. E. (1992). Conditional nonlinear planning. In Kaufmann, M., editor, *Proceedings of the First International Conference on Artificial Intelligence*, page 189–197, College Park, Maryland.

Richter, S. and Westphal, M. (2010). The Lama planner: guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 2(5):115–135.

Sanner, S. (2011). Relational dynamic influence diagram language (rddl): Language description. In *Proceedings of the Seventh International Planning Competion*.

Simon, H. and Newell, A. H. (1969). Gps: A case study in generality and problem solving. *Artificial Intelligence*, 2(5):109–124.

Simon, H. A. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118.

Teichteil-königsbuch, F., Infantes, G., and Kuter, U. (2008). Rff: A robust ff-based mdp planning algorithm for generating policies with low probability of failure. In *Proceedings of the sixth International Planning Competition*.

Wu, J.-H., Kalyanam, R., and Givan, R. (2008). Stochastic enforced hill-climbing. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, pages 396–403, Sydney, Australia.

Yoon, S. W., Fern, A., and Givan, R. (2007). Ff-replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, Providence, Rhode Island, USA.

Younes, H. L. S. and Littman, M. L. (2004). Ppddl1.0: An extension to pddl for expressing planning domains with probabilistic effects. *In Technical Report CMU-CS-04-162*.

Younes, H. L. S., Littman, M. L., Weissman, D., and Asmuth, J. (2005). The first probabilistic track of the international planning competition. *Journal of Artificial Intelligente Research*, 24:851–887.

Zettlemoyer, L. S., Pasula, H., and Kaelbling, L. P. (2005). Learning planning rules in noisy stochastic worlds. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 911–918.

Zickler, S. and Veloso, M. (2010). Variable level-of-detail motion planning in environments with poorly predictable bodies. In *Proceedings of the nineteenth European Conference on Artificial Intelligence*, Lisbon, Portugal.