Strong Cyclic Planning with Incomplete Information and Sensing

Luca Iocchi, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
E-mail: <iocchi,nardi,rosati>@dis.uniroma1.it

Abstract.
Incomplete information and sensing are needed in order to design agents that operate in domains where their information acquisition capabilities are restricted and the environment may evolve in unpredictable ways. These representation issues, that have been studied by the work on reasoning about actions, are also being addressed from a planning perspective. The aim of this paper is to present a language $\mathcal{KL}$ for expressing planning domains with incomplete knowledge and sensing and by providing a new technique for generating cyclic plans in such a framework. The basic feature of the representation is the notion of belief state which is characterized in terms of epistemic formulas, representing the knowledge of the agent about the environment. The proposed planning method deals with such belief states using propositional reasoning as a building block.

1 Introduction

In many different domains and application scenarios an autonomous reasoning agent must deal both with *incomplete information* about the environment and with *partial observability* i.e. limited capabilities to gather information. In particular, these assumption are required for robotic agents that must execute complex tasks in hostile environments, for which human control or supervision is either impossible or very expensive, as in space applications. Robots for space applications need the ability of reasoning on the limited knowledge they have, accomplishing actions to acquire new knowledge during task execution, and devising conditional plans (or contingent plans), that take into account the possibility of acquiring knowledge during task execution.

The recent literature on planning shows several contributions that aim at extending the classical planning domain by relaxing some of the underlying assumptions, specifically: *complete information* about the environment and *full observability*. In this extended setting, actions may have non-deterministic effects and it becomes possible to deal with sensing, i.e. knowledge acquisition. To this end, a substantial amount of work has been accomplished in order to develop models, languages and algorithms for planning in the presence of incomplete information and sensing (see for example (Weld *et al.* 1998; Bonet and Geffner 2000; Bertoli *et al.* 2001)).

However, the problem of reasoning with *incomplete information* has been also the subject of a large body of research on reasoning about actions, that has developed several approaches to reconstruct the agent's world representation after the execution of actions: Situation Calculus (Reiter 2001), A-languages (Gelfond and Lifschitz 1993), dynamic logics (Rosenschein 1981). In these formalisms, the representation of actions allows to specify not only the persistence of properties, but also sensing actions (Scherl and Levesque 1993; De Giacomo *et al.* 1997; Lobo *et al.* 1997; Iocchi *et al.* 2000; Thielscher 2000; Petrick and Bacchus 2002). In particular, in (Levesque 1996) sensing actions have been explicitly characterized as *knowledge-producing actions*.

The ability of a reasoning agent to generate plans in presence of incomplete knowledge and sensing requires a common view of the problem. The basic idea is to characterize the knowledge of the agent about the situation and to distinguish it from the actual world state, that cannot be completely perceived. However, the two approaches differ in the representation: i) a logic-based representation of the agent's knowledge in the first group (Scherl and Levesque 1993; De Giacomo *et al.* 1997; Lobo *et al.* 1997; Iocchi *et al.* 2000; Thielscher 2000; Petrick and Bacchus 2002); ii) specialized structures denoting sets of possible states (belief states) in the second one (Weld *et al.* 1998; Bonet and Geffner 2000; Bertoli *et al.* 2001).

On the basis of these two kinds of representations, different algorithms have been proposed for plan generation and plan verification. In both cases, the presence of sensing actions, that have different outcomes

depending on the state of the world during their execution, naturally gives rise to branches in the plan, introducing the notion of conditional plans or contingent plans.

The aim of this paper is to show that the use of a logic-based approach to contingent planning is both expressive from a representation viewpoint and may gain computational leverage. Specifically, it allows for considering more complex planning domains, in which the solutions are given in terms of *strong cyclic plans*, i.e. plans with loops whose termination conditions are checked at run-time, and that thus may not terminate. Notice that such a notion of plan has been commonly addressed in domains with complete knowledge (see for example (Giunchiglia and Traverso 1999)), while we believe that it is very relevant also in domains with incomplete knowledge and sensing, since it allows for synthesizing plans containing not only `if-then-else` structures, but also `while` loops.

In the paper, we propose a logic-based language for action representation, which is based on the representation of the knowledge of the agent through an epistemic logic. Moreover, we propose an algorithm for the generation of cyclic plans given the action representation. The formalization is illustrated through a pedagogical example and some preliminary experimental results of plan generation are discussed.

2 A Planning Language with Incomplete Information and Sensing

Dealing with incomplete information about the environment requires to distinguish the representation of what is true in the world from what the agent knows about the world. A common way to do this is to represent the agent's knowledge in terms of *belief states*. A belief state is a set of states that the agent considers possible in a particular situation. The agent thus does not have complete knowledge since it does not know exactly which is the actual state of the world at each moment.

Contingent planning can be modeled as a *"non-deterministic control problem over belief space"* (Bonet and Geffner 2000), as follows:

- A finite space $\mathcal{B}$ of belief states b over the set of states S.
- An initial situation given a the belief state b_0.
- A goal situation being a set of belief states B_G.
- A set of actions $\mathcal{A}$.
- A subset of actions $A(b) \subseteq \mathcal{A}$ for each belief state b, denoting the actions that are executable in b.
- A set of observations $O(a, b)$ for each execution of the action a from the belief state b.
- A transition function $f_a(b) = b_a^o$ that for every action $a \in A(b)$ non-deterministically maps a belief state b into a belief state b_a^o for each observation $o \in O(a, b)$.
- The observation $o_{a(b)} \in O(a, b)$ obtained after executing the action a in the belief state b.
- A cost function $c(a)$, expressing the cost of executing the action a.

A *plan* is a solution of this planning problem and can be represented as a graph, in which nodes are labeled with belief states b, edges are labeled with actions executed in the outgoing belief state $a(b)$, and every node b has successor nodes corresponding to b_a^o for every $o \in O(a, b)$, that are defined by the transition function $f_a(b)$. The graph has a single source node, that corresponds to the initial belief state b_0, and all the terminal nodes correspond to goal belief states, i.e. they belong to the set B_G.

Given this general framework that defines contingent planning, authors have attacked the problem from two different points of view: on the one end, there have been solutions to the general problem formulated as above, considering both acyclic and cyclic plans (e.g. (Hansen and Zilberstein 1998)), on the other hand, other authors focussed on defining languages for describing such problem in a more compact (although also restricted) way, and on the definition of efficient algorithms for solving a variant of the general problem (e.g. (Weld *et al.* 1998; Bonet and Geffner 2000; Bertoli *et al.* 2001; Petrick and Bacchus 2002)).

The approach presented in this paper follows the second line and it presents a language for representing the "non-deterministic control problem over belief space" in a compact, but also restricted way and an algorithm for efficient planning. With respect to other related approaches we introduce two novel aspects: 1) the representation is given in terms of the knowledge of the agent, as in (Petrick and Bacchus 2002), and it is significantly more efficient than the representation given in terms of possible worlds; 2) the generation of strong cyclic plans that allows for considering more complex domains and plans.

In the following of this section we describe a language for representing planning domains with incomplete knowledge and sensing (we call it $\mathcal{KL}$[1]), that is based on the ability to model the agent's knowledge.

[1]We have also defined a PDDL-like syntax for the language $\mathcal{KL}$, see (Iocchi *et al.* 2003).

Then in the next sections we present the algorithm for generating cyclic plans and experimental results.

2.1 The language $\mathcal{KL}$

The language $\mathcal{KL}$ makes use of epistemic formulas of the logic $\mathcal{ALCK}_{\mathcal{NF}}$ (see (De Giacomo *et al.* 1997; Iocchi *et al.* 2000) for details). More specifically, we introduce a set of primitive properties (or fluents) P, that will be used to characterize the possible states of the world. The primitive fluents $\mathcal{P}$ may either be either predicates or terms containing variables ranging over finite domains, in such a way that the set of belief states $\mathcal{B}$ remains finite. Notice that these variables are typically used only for describing domains in a more compact way, but do not increase the representation power of the language. For example, the term $in(x, y)$ with $x \in \{1, 2\}$ and $y \in \{1, 2\}$ can be replaced by the four predicates $in_{11}, in_{12}, in_{21}, in_{22}$. Also fluent with multiple finite values can be treated in the same way. Therefore, we can limit the description to the use of propositional formulas, since its extension to the case of terms with finite variables and fluents with multiple finite values is quite straightforward.

In our framework, a belief state b can be represented through an epistemic formula $\mathbf{K}\phi_b$, such that ϕ_b is a propositional formula, denoting the agent's knowledge about the world.

The language presented here allows for defining a restricted domain, with respect to the general one presented before, in the sense that not every belief states $b \in \mathcal{B}$ can be represented through epistemic formulas. In addition, as commonly done in reasoning about action, we restrict the formulas that are used for expressing the effects of an action to be a conjunction of literals, i.e. we do not allow epistemic disjunctive formulas (except for the sensing effects of an action) (De Giacomo *et al.* 1997; Iocchi *et al.* 2000). In fact, epistemic disjunction is used to model that after the execution of a sensing action a property is either known to be true or known to be false. This form of epistemic disjunction is specifically treated in our algorithm (see (De Giacomo *et al.* 1997; Iocchi *et al.* 2000)), while other forms of epistemic disjunction are not allowed. Notice also that this notion is different from that of noisy sensors that instead model the possibility of having a different result with respect to the actual state of the world. In this paper we will not deal with noisy sensors.

More specifically, referring to the general definition of the problem given above:

- The initial belief state $b_0 \in \mathcal{B}$ will be represented by the epistemic formula $\mathbf{K}\phi_{INIT}$.
- A goal B_G is the set of belief states $\{b \in \mathcal{B} \mid \mathbf{K}\phi_b \Rightarrow \mathbf{K}\psi_{GOAL}\}$.
- Given a set of precondition terms $pre_a : \mathbf{K}\alpha$ for the actions $a \in \mathcal{A}$, the subset of actions $A(b)$ applicable from the state b is $A(b) = \{a \in \mathcal{A} \mid \mathbf{K}\phi_b \Rightarrow \mathbf{K}\alpha\}$.
- The set of observations $O(a, b)$ are not modelled explicitly within our logical framework, but are considered as shown below.
- The transition function $f_a(b) = b_a^o$ is defined through the combination of: 1) deterministic effects of the form $post_a : \wedge_i(\mathbf{K}\alpha_i \rightarrow \mathbf{K}\beta_i)$ (i.e. a conjunction of generally conditional effects, in which we often simplify $\mathbf{K}\top \rightarrow \mathbf{K}\beta_i$ with $\mathbf{K}\beta_i$); 2) sensing effects of the forms: $sense_a : P$, where P is a fluent in $\mathcal{P}$, that will be known after the execution of a; 3) forgetting effects of the forms: $post_a : \neg\mathbf{K}P$, expressing that the fluent P will be unknown after the execution of the action a; 4) a set of static axioms that describe domain constraints to be applied in every belief state; 5) the default inertial propagation of all the fluents that do not affect consistency of the successor state. The resulting belief state b_a^o is represented by an epistemic formula that is build by appropriately computing the above effects. In case the action a has no sensing effects, then the successor belief state is unique, i.e. $|O(a, b)| = 1$, while in the case of a sensing effect on a fluent P we will have two successor belief states according to the two possible values of the observation of P, i.e. $|O(a, b)| = 2$. Obviously, this can be generalized to a combination of sensing effects on different fluents.
- The observation $o_{a(b)} \in O(a, b)$ obtained after executing the action a will thus be given by the knowledge of the value of a fluent P.
- In this paper we do not consider costs of the actions.

The above definition of the planning problem relies on the ability of expressing the axioms for defining the transition function $f_a(b)$. While other choices are possible, we present here a framework able to characterize in a very compact and effective way the dynamics of a system. Moreover, the ability of computing the successor belief state is a fundamental issue in our approach and further details on computing the epistemic formula characterizing such successor belief state are given in (De Giacomo *et al.* 1997; Iocchi *et al.* 2000).

3 Planning

In this section we address planning for the $\mathcal{KL}$ language. In particular, we define a planning algorithm

that is able to generate strong cyclic plans in such domains with incomplete knowledge and sensing. This allows for considering a larger set of domain problems for planning, namely all those problems for which a conditional solution does not exist, but that admit a strong cyclic plan.

The construction of the plan can be decomposed into two basic tasks: (i) the generation of successor belief states and (ii) the search for the plan in the belief state space. With respect to the first task, we can further decompose the problem into: verification of the precondition for action execution and computation of the effects, i.e. construction of the successor belief state.

In (De Giacomo *et al.* 1997; Iocchi *et al.* 2000) a solution to the problem of generating the successor belief (or epistemic) state is provided, by presenting an algorithm for constructing a complete representation of the belief states reachable from a given initial belief state. Such a representation is called First-Order Extension (FOE) and implements the epistemic reasoning that allows for dealing with transition between belief states, without considering explicitly the possible states included in these belief states. Moreover, the FOE is shown to provide a correct and complete representation of the set of belief states for the purpose of finding the plans for the given goal.

The algorithm for strong cyclic plan generation presented in this paper exploits our previous work on reasoning with epistemic states (De Giacomo *et al.* 1997; Iocchi *et al.* 2000), adding the ability of dealing with cyclic plans and to integrate heuristics in the search space that can significantly improve the computational performance of the planner.

The algorithm shown in Figure 1 takes as input a $\mathcal{KL}$ domain description Σ, an initial belief state b_0 and a description of the goal ψ_{GOAL}, and returns a plan (possibly cyclic) that is a solution of the planning problem as defined in Section 2. The plan is a graph represented by a set of tuples $< b_i, a_i, b_j >$, whose meaning is that from the belief state b_i it is possible to execute the action a_i leading to the successor belief state b_j. The action a_i can be either an action without sensing effects, in which case b_j is unique, or an action with sensing effects, in which the observation set $O(a_i, b_i)$ is given by two belief states $\{b_j, b'_j\}$, and thus both the tuples $< b_i, a_i, b_j >$ and $< b_i, a_i, b'_j >$ must be present in the plan.

The Plan Generation algorithm reported above uses the function $findLinearPath(\Sigma, b, \psi_{GOAL})$ that returns a linear path (a sequence of actions without cycles and branches) from the belief state b to a goal belief state in which ψ_{GOAL} holds. Since a linear path is a sequence of actions, in this function only one outcome of a sensing action is considered, and thus the returned path may have states that must be further expanded. The function $findLinearPath(\Sigma, b, \psi_{GOAL})$ "marks" those linear paths for which it was not possible a complete expansion, to avoid reconsidering them in the future. In other words, the function returns different paths if called multiple times with the same parameters.

Algorithm 1 Plan Generation

Procedure PLAN_GENERATION
Input: Domain Σ, initial belief state b_0, goal description ψ_{GOAL}
Output: Plan $\Pi = \{< b_i, a_i, b_j >\}$

$\Pi = findLinearPath(\Sigma, b_0, \psi_{GOAL})$
while $\Pi \neq \emptyset$ and $\exists < b_i, a_i, b_j > \in \Pi$, such that $O(a_i, b_i) = \{b_j, b'_j\}$ and $< b_i, a_i, b'_j > \notin \Pi$ **do**
  $\Pi' = PLAN\_GENERATION(\Sigma, b'_j, \psi_{GOAL})$
  if $\Pi' \neq \emptyset$ **then**
    $\Pi = \Pi \cup \{< b_i, a_i, b'_j >\} \cup \Pi'$
  else
    $\Pi = findLinearPath(\Sigma, b_0, \psi_{GOAL})$
  end if
end while
return Π

The ability to verify the equivalence of belief states through epistemic formulas is essential in the algorithm: 1) during the execution of the $findLinearPath$ function in order to guarantee that the returned path does not contain cycles; 2) in combining two plans in order to build graphs from the paths extracted, and thus introducing loops; 3) in determining that there are no solutions to the planning problem, when all the belief states have been examined.

As a difference with previous work done in this direction in (Petrick and Bacchus 2002), our formalism exploits this capability to provide a *sound and complete* method for generation of both conditional plans and strong cyclic plans.

Finally, the termination of the algorithm is a consequence of the following observations: i) the complete expansion of a single path initially generated by $findLinearPath$ terminates since the algorithm expands each state only once for every graph generated during the process; ii) the number of paths generated by the different calls of $findLinearPath$ is finite.

4 Example and Experimental Results

In this section, we describe an example domain with the aim of highlighting the features of the proposed language and of the planning algorithm. Some experimental results also show the feasibility of the approach and a preliminary comparison with related approaches.

4.1 Mars Rock Analysis Example

The following example describes a domain in the language $\mathcal{KL}$, that models the high-level activities of a rock analysis task for a Mars Exploration Rover Mission (see for example (Washington *et al.* 1999)). Since we are mainly concerned in the high-level control of the rover, this domain does not consider some important characteristics of a real domain, such as temporal duration of actions, resource availability, etc. Moreover, in several domains the control of an autonomous vehicle is very complex and it is usually preferred to design an architecture that provides different levels of controls, rather then integrating all of them in a single framework. Furthermore, high-level planning is also very useful during the development stage, in which it is important to devise and test the basic functionalities that must be implemented in an autonomous vehicle in order to accomplish its tasks.

To this end, we suppose that the Mars Rover has the following basic capabilities:

- localization and navigation ability that allows for reliably mapping the area to explore, representing it through a rectangular $n \times m$ grid;
- the ability of safely move through the grid: this will be modeled by the high-level actions `up`, `down`, `left`, and `right`;
- a sensing ability for detecting if, in the cell of the grid where the rover is located, there is a new rock to be analyzed: this is modeled through the high-level action `search_new_rock`;
- the ability of getting close to the discovered rock, and of collecting data from it, labeling this rock in such a way that a subsequent execution of the `search_new_rock` action will not find it again: we express this ability through the high-level action `analyze_rock`.

The description of this domain in the language $\mathcal{KL}$ is reported in Table 1. Given such a description, we may specify a planning problem whose goal is to explore all the mapped area in order to analyze all the detected rocks. Thus, starting from an initial state in which only the starting position of the rover in the grid is known, denoted for example by $\mathbf{K}\phi_{INIT} = \mathbf{K}in(1,1)$, we may define the goal expressing that in every cell of the grid there must be no rocks to be analyzed. In the defined language the goal is expressed by the formula $\mathbf{K}\psi_{GOAL} = \mathbf{K} \wedge_{i,j} \neg new\_rock(i,j)$, where (i,j) denotes a cell of the grid.

The plan extracted by the planner for this problem is the one that allows the rover to reach all the cells in the grid (through the motion actions) and to execute there a loop for analyzing all the detected rocks one after the other (see Figure 1 in which the grid size is 2×2).

Notice that, there are several `while` loops in the plan (one for each cell of the grid), in which the rover performs a number of analysis actions until the condition checking for new rocks to be analyzed becomes false. As already mentioned, the presence of loops in the plan is due to the capability of our representation language to establish when two belief states are equivalent. In fact, after executing an action `analyze_rock`, the belief successor state is identical to the one before performing the sensing action `search_new_rock`, since the status of the agent's knowledge is that it does not know whether there are rocks to be analyzed.

4.2 Experimental results

We have performed some tests to evaluate the behaviour of our planner both on the example reported above and on a few other examples taken from the literature. In particular, we have integrated within the $findLinearPath$ function a heuristic depth-first search. The experimental results for this example have shown that the implemented heuristic depth-first search allows for a linear computational cost for generating linear-sized plans, even though the search space is obviously exponential in the size of the domain description.

We have compared the implementation of our planner, $\mathcal{K}$-Planner, with related planners that have similar functionalities and representation capabilities: PKS (Petrick and Bacchus 2002), that makes use of a modal logic based representation of the agent's knowledge, and MBP (Bertoli *et al.* 2001), that instead represent knowledge in terms of sets of possible states. For this comparison we have chosen domains (e.g. Medical (Weld *et al.* 1998), Ring (Bertoli *et al.* 2001), etc.) that do not contain cyclic solutions, since the mentioned planners cannot deal with cyclic plans. From the analysis on the performance of our algorithm in such problems, we found out that the HDF search in our algorithm is very efficient when the structure of the solution comprises a linear concatenation of similar blocks, as in the reported cases. In other words, when a linear path is initially found

Action a	pre_a	$post_a$	$sense_a$
`up`	$\wedge_j \mathbf{K}\neg in(n,j)$	$\mathbf{K}in(i,j) \rightarrow \mathbf{K}in(i+1,j)$	
`down`	$\wedge_j \mathbf{K}\neg in(1,j)$	$\mathbf{K}in(i,j) \rightarrow \mathbf{K}in(i-1,j)$	
`left`	$\wedge_i \mathbf{K}\neg in(i,1)$	$\mathbf{K}in(i,j) \rightarrow \mathbf{K}in(i,j-1)$	
`right`	$\wedge_i \mathbf{K}\neg in(i,m)$	$\mathbf{K}in(i,j) \rightarrow \mathbf{K}in(i,j+1)$	
`search_new_rock`	$\mathbf{K}in(i,j)$		$new\_rock(i,j)$
`analyze_rock`	$\mathbf{K}in(i,j) \wedge \mathbf{K}new\_rock(i,j)$	$\neg\mathbf{K}new\_rock(i,j)$	

Table 1: Rock Analysis domain description

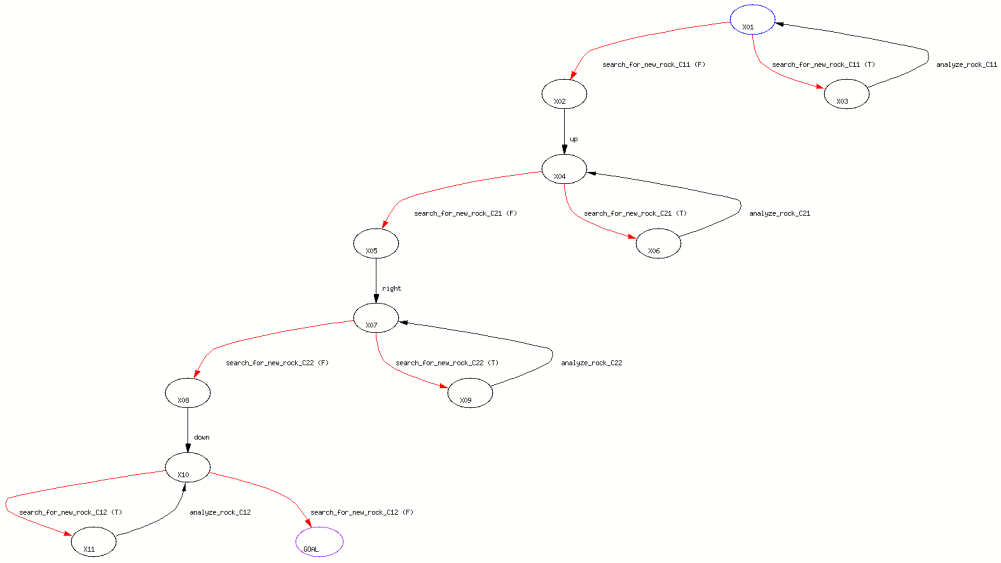

Figure 1: MER rock analysis plan

to reach the goal (here the one in which all the sensing actions will return false, i.e. no rocks are found in the cell), then the completion of this partial solution is achieved very quickly.

This behaviour confirms the results reported in (Petrick and Bacchus 2002), showing that the ability of explicitly representing the knowledge of the agent allows for a more efficient implementation of a planner, with respect to those implementations that model the knowledge in terms of sets of possible states (e.g. (Weld *et al.* 1998; Bonet and Geffner 2000; Bertoli *et al.* 2001)) In fact, in the considered problems, planners based on the explicit representation of the agent's knowledge ($\mathcal{K}$-Planner and PKS) can easily deal with larger domains as opposed to those planners (e.g. MBP) that use sets of possible states (see (Bertoli *et al.* 2001) for a comparison among these planners). As an example, we report in Table 2 the performance in the medical domain (we have obtained similar data for the ring domain) of the three planners considered. The numerical data for PKS are taken by their paper (Petrick and Bacchus 2002), while data for $\mathcal{K}$-Planner and MBP are taken by running them on the same 2.5 GHz CPU. The data reported here are not meant to be an efficiency comparison between our planner and PKS, but rather to confirm that the use of explicit representation of knowledge of the agent in planning domains with sensing and incomplete information is significantly more efficient than approaches based on possible worlds.

Finally, as already mentioned, our language allows for a more compact representation of a domain, but is not able to deal with some specific representation of planning domains (namely the ones that require reasoning by cases). However, this limitation in the representation power is not very restrictive from a representational viewpoint (in fact, we can represent all the example domains given in (Weld *et al.* 1998; Bonet and Geffner 2000; Bertoli *et al.* 2001; Petrick and Bacchus 2002)), while providing substantial computational advantages, by ruling out some forms of reasoning by cases.

5 Conclusions

In this paper, we have proposed $\mathcal{KL}$, a logic-based language for action representation, which allows for considering complex planning domains with incomplete knowledge and sensing actions. In such scenarios, we have discussed the need of cyclic plans in order to solve the contingent planning problem, and we have presented an algorithm for the generation of cyclic plans for a given domain specification. Moreover, some preliminary experimental results of plan generation have been discussed.

In $\mathcal{KL}$ one can express the most important forms of sensing actions and incomplete information proposed in the recent planning literature: e.g., (Weld *et al.* 1998; Bertoli *et al.* 2001; Petrick and Bacchus 2002). Furthermore, (Petrick and Bacchus 2002) presents an approach to planning in the presence of sensing based on the explicit representation of the epistemic state of the agent, through the use of a very expressive first-order modal epistemic formalism. Epistemic states are logically formalized by knowledge bases (i.e., sets of formulas) in such a logic, actions are specified by means of updates (deletions and insertions) over such knowledge bases, hence the semantics of transitions between epistemic states is expressed at a meta-logical level. However, the main difference between their method and the one proposed in this paper, is that we provide a *sound and complete* planning algorithm, able to generate both conditional plans and strong cyclic plans.

The present work can be extended in several directions. In particular, we are currently working on extending our framework in order to allow for the representation of both qualitative and quantitative uncertainty in the effects of actions.

Acknowledgments

This work is partially supported by ASI (Italian Space Agency) under project ARISCOM (Contract I/R/215/02).

References

P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 473–478, 2001.

B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of Int. Conf. on AI Planning and Scheduling (AIPS'00)*, pages 52–61, 2000.

Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing for a mobile robot. In *Proc. of 4th European Conference on Planning (ECP'97)*, 1997.

M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *Proc. of the 5th Eur. Conf. on Planning (ECP'99)*, 1999.

E. Hansen and S. Zilberstein. Heuristic search in cyclic AND/OR graphs. In *Proc. of AAAI-98*, pages 412–418, 1998.

Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical

Illnesses	$\mathcal{K}$-Planner	PKS (Petrick and Bacchus 2002)	MBP (Bertoli *et al.* 2001)
4	1	-	10
6	2	-	300
8	4	-	11500
10	8	-	?
20	65	80	?
50	620	1610	?
100	4620	20390	?

Table 2: Medical domain - Search time in ms

framework and implementation. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 678–689, 2000.

L. Iocchi, D. Nardi, and R. Rosati. Strong cyclic planning with incomplete information and sensing. Technical Report 16-03, Dipartimento Informatica e Sistemistica Università di Roma La Sapienza, 2003.

Hector J. Levesque. What is planning in presence of sensing? In AAAI Press/The MIT Press, editor, *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 1139–1149, 1996.

J. Lobo, G. Mendez, and S. Taylor. Adding knowledge to the action description language $\mathcal{A}$. In *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI'97)*, pages 454–459, 1997.

R. P. A. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of Sixth International Conference on AI Planning and Scheduling (AIPS2002)*, 2002.

R. Reiter. *Knowledge in action: Logical foundations for describing and implementing dynamical systems*. MIT Press, 2001.

S. Rosenschein. Plan synthesis: a logical perspective. In *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, 1981.

Richard Scherl and Hector J. Levesque. The frame problem and knowledge producing actions. In *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI'93)*, pages 689–695, 1993.

Michael Thielscher. Representing the knowledge of a robot. In *Proc. of the International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pages 109–120, 2000.

R. Washington, K. Golden, J. Bresina, D.E. Smith, C. Anderson, and T. Smith. Autonomous rovers for mars exploration. In *Proc. of the 1999 IEEE Aerospace Conference.*, 1999.

D. S. Weld, C. R. Anderson, and D. E. Smith. Extending graphplan to handle uncertainty & sensing actions. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, 1998.