GENERAZIONE DI PIANI CICLICI CON CONOSCENZA INCOMPLETA E PERCEZIONE

GENERATION OF STRONG CYCLIC PLANS WITH INCOMPLETE INFORMATION AND SENSING

Luca Iocchi, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
E-mail: <iocchi,nardi,rosati>@dis.uniroma1.it

SOMMARIO/*ABSTRACT*

Considerare conoscenza incompleta e percezione è fondamentale per la realizzazione di agenti che operano in domini dove le proprie capacità di acquisizione delle informazioni sono limitate e l'ambiente può evolvere in maniera non prevedibile. Questi aspetti di rappresentazione sono stati studiati sia dai lavori nel campo del ragionamento sulle azioni, sia dal punto di vista dei sistemi di pianificazione automatica. Lo scopo di questo articolo è di presentare un linguaggio $\mathcal{KL}$ in grado di descrivere domini di pianificazione con conoscenza incompleta e percezione e di fornire una nuova tecnica per la generazione di piani ciclici in condizioni di osservabilità parziale per tale formalismo.

Dealing with incomplete information and sensing is 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 have been studied both by the work on reasoning about actions and 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 under partial observability in such a framework.

Keywords: Planning under uncertainty, Conditional planning

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 rescue operations, space applications, etc. Such robots need thus 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 [16, 2, 1]).

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 [12], A-languages [4], dynamic logics [13]. In these formalisms, the representation of actions allows to specify not only the persistence of properties, but also sensing actions [14, 9, 3, 10, 8, 15, 11].

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 [14, 9, 3, 10, 8, 15, 11]; ii) specialized structures denoting sets of possible states (belief states) in the second one [16, 2, 1].

On the basis of these two kinds of representations, dif-

ferent 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 runtime, and that thus may not terminate. This notion of plan has been commonly addressed in domains with complete knowledge (see for example [5]), 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, and an algorithm for the generation of cyclic plans under partial observability. The formalization is illustrated through a pedagogical example and 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 which is the actual state of the world.

Contingent planning can be modeled as a *"nondeterministic control problem over belief space"* [2], 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 $o \in O(a, b)$ 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. [6]), 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. [16, 2, 1, 11]).

The approach presented in this paper follows the second line and it presents a language for representing the "nondeterministic 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 [11], and it is significantly more efficient than the representation given in terms of possible worlds; 2) the generation of strong cyclic plans under partial observability, 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. 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 [3, 8] 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 be either predicates or terms containing variables ranging over finite domains,

[1] We have also defined a PDDL-like syntax for the language $\mathcal{KL}$, see [7].

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 fluents 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.

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) [3, 8]. 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 [3, 8]), 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:

- The initial belief state $b_0 \in \mathcal{B}$ is represented by an 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 $o \in O(a, b)$ 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 planning problem relies on expressing the logical axioms for defining the transition function $f_a(b)$. While other choices are possible, we have used a framework able to characterize in a very compact and effective way the dynamics of a system. The ability of computing the successor belief state is fundamental in our approach and further details are given in [3, 8].

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 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 [3, 8] 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 [3, 8], 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.

Algorithm 1 Plan Generation

Procedure PLAN_GENERATION$(\Sigma, b'_j, \psi_{GOAL}, \mathcal{S})$
Input: Domain Σ, initial belief state b_0, goal description ψ_{GOAL}, set of states in the current plan S (initially empty).
Output: Plan $\Pi = \{< b_i, a_i, b_j >\}$

$\Pi =$ findLinearPath$(\Sigma, b_0, \psi_{GOAL}, \mathcal{S})$
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**
  $\mathcal{S}' = \mathcal{S} \cup \{b_i | < b_i, \cdot, \cdot > \in \Pi \vee < \cdot, \cdot, b'_j > \in \Pi\}$
  $\Pi' =$ PLAN_GENERATION$(\Sigma, b'_j, \psi_{GOAL}, \mathcal{S}')$
  if $\Pi' \neq \emptyset$ **then**
    $\Pi = \Pi \cup \{< b_i, a_i, b'_j >\} \cup \Pi'$
  else
    $\Pi =$ findLinearPath$(\Sigma, b_0, \psi_{GOAL}, \mathcal{S})$
  end if
end while
return Π

The Plan Generation 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 algorithm is based on the idea of finding linear paths from an initial belief state to a goal state (without considering multiple effects of sensing actions), and then completing these paths by considering the different effects of sensing. To this end, we make use of a set of states $\mathcal{S}$, containing the belief states considered in the plan under construction. Since, during the execution of the algorithm, a partial plan contains all paths that lead to goal states, the set $\mathcal{S}$ always contains belief states from which it is possible to reach a goal state. Therefore, during the search for a linear path, if there exists a path that leads to a belief state in $\mathcal{S}$, then this path can be chosen and the search interrupted since from the subsequent belief state a path to a goal state has already been computed. This mechanism not only prevents to repeat the same computation at different times, but also allows for generating direct acyclic and cyclic plans.

More specifically, this algorithm uses the function findLinearPath$(\Sigma, b, \psi_{GOAL}, \mathcal{S})$ that returns a linear path (a sequence of actions without cycles and branches) from the belief state b to a goal belief state, that is either a state in which the goal ψ_{GOAL} holds or a state included in the set $\mathcal{S}$. The search strategy used by this function can be implemented in different ways. In our implementation, called $\mathcal{K}$-Planner, we have used an heuristic search based on a simple metric that measures the "distance" of a belief state from the goal.

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. This is taken into account by the multiple recursive calls of the function in the loop, that is executed until either the plan is complete (and hence a solution has been computed) or the plan returned by findLinearPath is empty (thus the solution does not exist). Moreover, the function findLinearPath$(\Sigma, b, \psi_{GOAL}, \mathcal{S})$ "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.

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 [11], our formalism exploits this capability to provide a *sound and complete* method for generation of both conditional plans and strong cyclic plans.

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 Experiments in Cyclic Plan Generation

In this section, we describe a planning domain for which a conditional solution does not exists. More specifically, the goal is not guaranteed to be reached, unless some actions are repeated more times. This kind of domains thus admit only a strong cyclic solution, that is a plan containing loops, that when executed it may not terminate and if it terminates then the goal is achieved.

The cleaning domain (see Table 1) describes a world in which a robot is involved in a cleaning task in an office-like environment, that is formed by a number of different rooms $\mathtt{R}_i$ connected through a corridor C. In the environment

Action a	pre_a	$post_a$	$sense_a$
`enter_Rj`	$\mathbf{K}in(C)$	$\mathbf{K}in(R_j)$	
`exit_Rj`	$\mathbf{K}in(R_j)$	$\mathbf{K}in(C)$	
`takeout_Rj`	$\mathbf{K}in(R_j) \wedge \mathbf{K}obj\_in(R_j)$	$\mathbf{K}in(C) \wedge \neg\mathbf{K}obj\_in(R_j)$	
`scan_Rj`	$\mathbf{K}in(R_j)$		$obj\_in(R_j)$

Table 1: Cleaning domain description

there are some objects in unknown positions and quantities in the rooms and the robot has sensing capabilities for identifying such objects. The robot can move through this environment (`enter`, `exit` in the rooms), move objects out of a room (`take_out`), and search for objects in the rooms (`scan`).

Notice that the fact that an unknown number of objects can be present in a room is modelled through the effect of the action `take_out`, expressing that after an object is taken out from a room, it is not known if the room is clean (i.e. if there are other objects).

The planning problem defined for this domain is defined by $\phi_{INIT} = \mathbf{K}in(C)$ and $\psi_{GOAL} = \wedge_i \mathbf{K}\neg obj\_in(R_i)$, in which the robot knows to be in the corridor in the initial state (no knowledge about objects in the rooms is given), and the goal is reached when all the rooms have been cleaned. Such a domain does not admit any valid conditional plan, unless loops are considered. In fact, due to the presence of an undefined number of objects in every room the action `take_out` may need to be executed more times in order to achieve the goal. The strong cyclic plan generated by our $\mathcal{K}$-Planner (when considering 3 rooms in the environment) is reported in the Figure 1. The robot will examine one room at a time and will take out from it all the objects. The goal is achieved when all the three rooms have no objects inside.

The computational properties of our algorithm allows for a good scalability in the size of the domain. In fact, the algorithm is based on first finding a linear partial solution: in the example the linear path from the initial state to the goal considering all sensing results as *false* will be returned first. Then from this linear solution a completion is computed for every sensing: in the example the three loops are added on the *true* branches of sensing actions. It is important to highlight here that the ability of computing equivalence of belief states allows (when possible) for generating only a few action steps to form a loop. In the above example, when the sensing action is *true* only the linear path (`take_out_Rj`; `enter_Rj`) is computed by the algorithm and merged to the current partial plan. In this way with increasing number of rooms, once the first linear path is found (this can be done with different heuristics), then the completion of the plan is usually performed very quickly.

In Fig. 2 we summarize computational time for the Cleaning domain, distinguishing time needed for finding the first linear path (second column) from time required for completing this path generating the strong cyclic plan (third column). Notice that the completion of the first linear path is substantially linear in the size of the domain, while the generation of the first linear path obviously is not. The presented algorithm shows very good performance in those domains in which the solution is composed by: i) a first linear path obtained with a particular configuration of sensing results, ii) several paths that consider all the other sensing results that are connected to the first linear path or anyway reach the goal with a limited number of steps.

5 Experiments in Conditional Planning

We have performed some tests to evaluate the behaviour of our planner also on some standard domains for conditional planning and compared the performance of our planner with well-known similar approaches: PKS [11], that makes use of a modal logic based representation of the agent's knowledge, and MBP [1], that instead represent knowledge in terms of sets of possible states.

The results of these experiments show that the heuristic 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. As already discussed, in our algorithm when a linear path is initially found to reach the goal, then the completion of this partial solution is usually achieved very quickly. However, there are situations in which the algorithm does not present this nice computational property: namely, those domains such that, for every path to be completed, it is necessary to generate a new path that cannot be connected to the previous generated paths and whose length is comparable with the other paths. In other words, in those problems in which the solution cannot be represented as a DAG, but instead must be represented as an almost complete tree, then the proposed algorithm does not present the benefits described above.

The behaviour shown in this paper also confirms the results reported in [11], 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. [16, 2, 1]) 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 [1] for a comparison among these planners). As an example, we

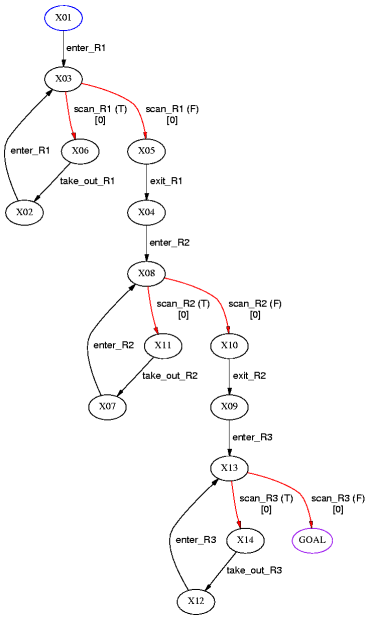

Figure 1: Strong cyclic plan for cleaning domain

Rooms	First Linear Path	Complete Linear Path
10	60	3
20	240	8
30	1040	35
40	3230	60
50	7900	85

Figure 2: Cleaning domain search (ms)

Illnesses	$\mathcal{K}$-Planner	PKS [11]	MBP [1]
4	1	-	10
6	2	-	300
8	4	-	11500
10	8	-	?
20	65	80	?
50	620	1610	?
100	4620	20390	?

Figure 3: Medical domain search (ms)

report in Fig. 3 the performance in the medical domain[2].

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). For example, the Ring domain, as modeled in [1], requires reasoning by cases in order to generate a plan in which the robot visits all the rooms connected in a ring without knowing and sensing its initial location. In our formalism, we have to model the domain by introducing a sensing action the expresses the capability of the robot to localize itself in the environment. The Ring domain with Localization capabilities considered in our framework has computational properties similar to the medical domain and thus computational performance with respect to the original Ring domain in MBP is similar to the one shown in Fig. 3.

Summarizing, our formalism provides for a more efficient planning method in those domains that do not require reasoning by cases. Notice also that this limitation in the representation power is not very restrictive from a representational viewpoint (in fact, we can represent almost all the example domains given in [16, 2, 1, 11]), while it provides substantial computational advantages.

6 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 strong cyclic plans and some experiments in plan generation.

In $\mathcal{KL}$ one can express the most important forms of sensing actions and incomplete information proposed in the recent planning literature: e.g., [16, 1, 11]. Similarly to our approach, in [11] the authors presents a method for 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.

[2]The value for PKS are taken by their paper [11], while $\mathcal{K}$-Planner and MBP are evaluated on the same 2.5 GHz CPU.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

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

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

[7] 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.

[8] Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 678–689, 2000.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.