Generation and execution of partially correct plans in dynamic environments

Alessandro Farinelli, Giorgio Grisetti, Luca Iocchi, Daniele Nardi and Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università "La Sapienza", Roma, Italy

Abstract

In this paper we present the recent developments of the approach to the design of Cognitive Robots (i.e. robots whose actions are driven by a formally developed theory of action), that are capable of performing tasks in a coordinated way. The logic of actions that we adopt is an epistemic dynamic logic, where it is possible to derive acyclic branching plans (branches corresponding to sensing actions), including primitive parallel actions.

In the present work, we consider an extended notion of plan by admitting a simple class of cycles that arise from the attempt to recover from the failure states originated by sensing actions. The proposed extension allows us to address the problem of generating plans that handle a form of synchronization based on the recognition of specific situations through sensing actions, including forms of coordination required in a multi-robot scenario.

Introduction

The focus on planning from a Cognitive Robotics perspective is to provide the robot with plans that are driven by an action theory and can be effectively executed by *robotic agents*.

A plan can be viewed as a sequence of actions that are to be executed by the robot, or as complex programs that include loops, nondeterministic choice and conditional constructs (see for example (Reiter 2001)). In the former case, plans can be generated by a planner (De Giacomo *et al.* 1996; Reiter 2001) in the latter case the system is given the plan and computes the consequences of the execution of the plan (Reiter 2001).

In fact, attempts have been made to extend the plan generation approach by developing systems that take into account non determinism (Cimatti *et al.* 1997), sensing (De Giacomo *et al.* 1997), or concurrent actions (Iocchi, Nardi, & Rosati 2000) in the plan generation process.

The work on classical planning (where the plan is always generated by the system) in the last years has been focusing on planners that take into account non-deterministic actions, but rely on the hypothesis of complete observability (Giunchiglia & Traverso 1999; Jensen, Veloso, & Bowling 1999; Blum & Furst 1995). In the presence of non-determinism, it is relevant to consider, in addition to the plans that are classically guaranteed to achieve the goal, also plans that may fail depending on the success/failure of the actions being executed. Consequently, different (and rich) notions of plan have been developed in order to account for branches and cycles in the plan structure (see for example (Giunchiglia & Traverso 1999)). These approaches are based on the assumption that during the execution of the plan the robot has complete observability on all the properties that are considered in the dynamic system formalization. However, this assumption may not be valid in some real applications for mobile robots.

More recently, in (Bertoli *et al.* 2001) the hypothesis of complete observability is relaxed by introducing sensed properties and addressing the problem of generating tree-structured plans, where the branching structure is due to the sensing and incomplete knowledge is represented by belief states.

In this work we present a technique for generating plans containing a simple form of cycles, corresponding to the repeated execution of plans of a sensing action. This structure has been used for implementing synchronization among plans executed on different robots based on the recognition of a sensed condition.

The basis of the present work is the formal framework and the planning algorithm that generates acyclic branching plans (branches corresponding to sensing actions) described in (Iocchi, Nardi, & Rosati 2000). In the present work, we adopt a weaker notion of plan and provide a technique for generating plans possibly including a simple form of cycles that arise from the attempt to recover from the failure states originated by sensing actions.

The original motivation for devising such plans was the attempt to achieve some form of synchronization between robots in the multi-robot scenario provided by the RoboCup competitions (Kitano *et al.* 1998), more specifically the Sony Legged Robot League. Consequently, we address the system architecture that allows for the generation and execution of the plans on our *robotic agents* (Castelpietra *et al.* 2001), discuss the proposed approach and its implementation in the RoboCup scenario.

The paper is organized as follows. We first describe the

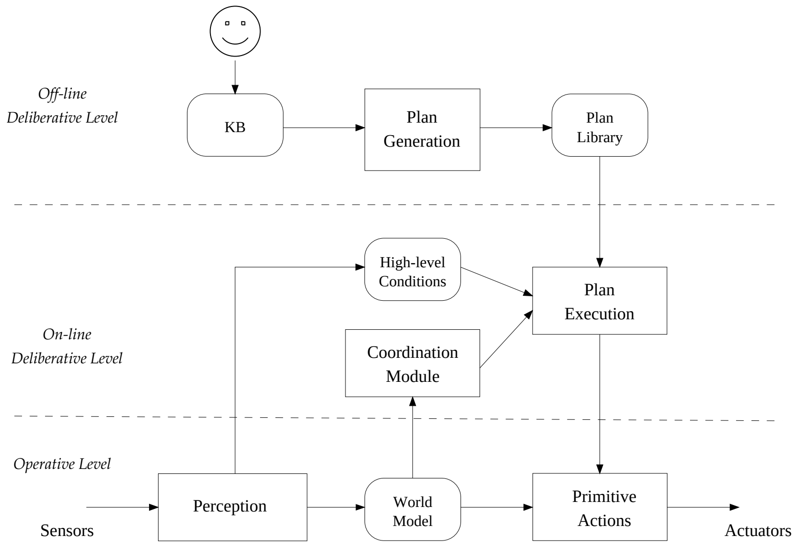


Figure 1: Layered architecture for our robots

architecture of our cognitive robots, then we recall the basics of our logical framework, we introduce the notion of *partially strong plans* and describe the implementation of the plan generation procedure. Finally, we provide some examples of plans that have been devised for coordination in the RoboCup Sony Legged environment.

System architecture

In this section we recall the layered hybrid architecture used for our cognitive mobile robots (see also (Iocchi 1999)) displayed in Fig. 1, that has been implemented on several different kinds of robotic platforms, namely Sony AIBOs, Pioneer, and home-made wheeled robots.

The *Operative Level* of the architecture is based on a numeric representation of the information acquired by the robot sensors and of the data concerning the current task, while the *Deliberative Level* is based on a symbolic representation of the information acquired by the robot sensors and of the data concerning the task to be accomplished: the *On-Line Deliberative SubLevel* is in charge of evaluating data during the execution of the task, while the *Off-line Deliberative SubLevel* is executed off-line before the actual task execution.

The deliberative level relies on a representation of the robot's knowledge about the environment. This knowledge is formed by both a general description of the environment provided by the robot's designer and the information acquired during task execution. In particular, the world model of this level contains symbolic information corresponding to the data in the geometric world model of the operative level.

The deliberative level is formed by three main components:

1. *Plan Execution Module* that is executed on-line during the accomplishment of the robot's task and is responsible for executing a plan by coordinating the primitive actions of a single robot;
2. *Coordination Module* that is responsible for assigning tasks to the robots in the team according to the current situation;
3. *Plan Generation Module*, that is executed off-line before the beginning of the robot's mission, and generates a set of plans to deal with some specific situations.

The Plan Execution Module is in charge of executing a plan, represented as a graph as described in the next section.

During the execution of a plan, this module checks for the conditions that guarantee the applicability of the current plan in the current situation. If the current plan is no longer executable, because some properties do not hold anymore, this module communicates this fact to the coordination module that will select another task and consequently another plan to be assigned to the robot.

Coordination among robots is achieved by a coordination module that selects which task must be accomplished by every robot, by sending to the Plan Execution Module the appropriate plan in the library that relates to this task. The coordination module is described in (Castelpietra *et al.* 2000) and it is based on a coordination protocol for dynamic assignment of roles and on the evaluation of *utility functions* computed by each robot for determining an estimation of its own capability to reach a given goal. The computation for the coordination protocol is distributed and the protocol is robust because it relies on a little amount of transmitted data. Each robot has the knowledge necessary to play any role, therefore robots can switch roles on the fly, as needed. In practice a role corresponds to a goal to be achieved by a robot and dynamic role exchange is a very effective way to select the robot that is in a better situation to accomplish a task. Moreover, when the situation at hand is such that the goal cannot be accomplished the coordination module will assign this task to another robot. This is obtained by decreasing the utility function for this role, so that the coordination protocol will eventually select another robot to perform this task.

The reasoning system processes a knowledge base containing a description of the actions to be performed by the robot using an action representation language, that is described in the next section. The Plan Generation Module implements an automatic planner that generates a plan to achieve a given goal in a specific situation, as described in the next section. Plan generation is accomplished off-line for efficiency reasons and it is achieved by restricting the situations of interest.

Plan Representation and Generation

In this section we first recall our formal framework for the specification of the dynamic system, and then present our new method for cyclic conditional plan generation, based on a notion of *partial correctness* of plans that allows for generating plans corresponding to programs with a general structure (ramifications and indefinite cycles).

Logical framework for dynamic system representation

We first briefly recall the main features of our formal framework for representing actions (we refer to (Iocchi, Nardi, & Rosati 2000) for a detailed presentation of the framework).

Generally speaking, our work on the definition of the logical framework is based on the idea of using a nonmonotonic formalism in order to logically reconstruct a number of expressive features for action representation. More specifically, we have analyzed the modal nonmonotonic description logic $\mathcal{ALCK}_{\mathcal{NF}}$ (Donini, Nardi, & Rosati 2002), and experimented its adequacy in the formalization of dynamic systems. It can be shown that such a formalism is able to logically capture and extend several formal frameworks for action representation (e.g., STRIPS and the language $\mathcal{A}$).

In our $\mathcal{ALCK}_{\mathcal{NF}}$ framework, we represent the dynamic system through a logical theory (called $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base). In such a theory, actions are represented by means of binary relations (roles), states are represented by means of domain elements (individuals), and state properties by means of unary predicates (concepts). Actions are represented using preconditions and effects. Preconditions are the conditions that are necessary for activating the action and indicate what must be true before the action is executed: they specify circumstances under which it is possible to execute an action. Effects are the conditions that must hold after the execution of the action and characterize how the state changes after the execution of the action.

In (Iocchi, Nardi, & Rosati 2000) we have shown that such a framework allows for expressing the following features:

- *ordinary* actions, i.e., actions whose effect is deterministic, although context-dependent;
- *sensing* (or knowledge-producing) actions, namely actions which allow the robot to know the value of a property in the current state of the world. The peculiarity of such actions lies in the fact that their execution only affects the robot's knowledge about the world, without changing the state of the external world;
- *concurrent execution* of single actions (both ordinary and sensing actions);
- *state constraints* (Lin & Reiter 1994), that is, ordinary first-order sentences expressing static relationships between dynamic properties, and *causal* dependencies between properties (Mc Cain & Turner 1995; Thielscher 1997). Both ordinary state constraints and casual constraints enforce *ramifications*, e.g. indirect effects of actions;
- several forms of persistence of properties. In particular, through the use of both monotonic and nonmonotonic solutions to the frame problem, we are able to formalize both inertial/non-inertial properties and inertial/non-inertial actions. We remark that we are interested in dynamic environments in which some properties may change due to events (usually called *exogenous events*) that cannot be predicted by the robot.

Due to the form of the assertions used in the formalization of the dynamic system, the interpretation structures (models) of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base can be interpreted as *transition graphs*. In particular, individuals represent states of the system and are labeled by concepts representing the properties (or fluents) that hold in that state; edges between individuals represent transitions between system states, and are labeled by roles representing the actions that cause the state transition. More specifically, each node (individual) denotes a different *epistemic* state of the robot, i.e., what the robot knows about the world: an edge labeled by an action R connects two such states (individuals) s, s' if the execution of R, when the robot's epistemic state is s, changes its epistemic state to s'.

As illustrated in (Iocchi, Nardi, & Rosati 2000), the set of models of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base Σ formalizing a dynamic system can be represented by means of a unique transition graph, called *first-order extension (FOE)* of Σ, which represents all the possible evolutions of the dynamic system. For instance, Figure 2 displays some examples of portions of FOEs. Observe that the nodes of these graphs represent the *epistemic states* of the robot, thus containing the properties that the robot *knows* about the environment, and not what is true in the environment. An *epistemic state* actually corresponds to a set of states in the world and thus allows for representing partial knowledge of the robot. Moreover actions here represent transitions between *epistemic states* (i.e. sets of states in the world). This is an important difference with respect to the graphs used in classical planning (Giunchiglia & Traverso 1999; Jensen, Veloso, & Bowling 1999; Blum & Furst 1995), where every node correspond to a state in the world with complete knowledge of the properties in the environment and actions maps world states to world states.

The notion of plan

The classical formulation of the notion of plan corresponds to the following: given the specification of

- the dynamic system
- the initial state
- a goal (property) G

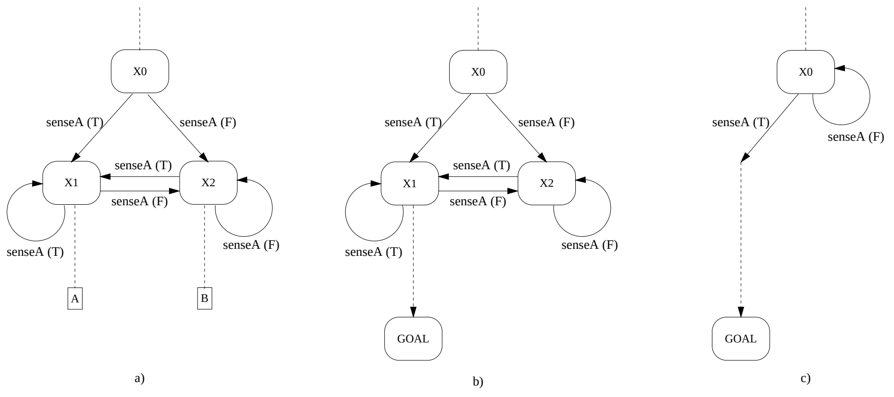


Figure 2: Plan structure for cyclic sensing actions.

a *plan for G* is a sequence of actions $\mathcal{S}$ such that the execution of $\mathcal{S}$ leads to a state s in which the goal G holds.

In order to have an effective notion of plan in our framework, we have to modify the above notion in many respects. In particular, the execution of a sensing action in a given state may have in general multiple outcomes. Hence, the presence of sensing actions requires to reformulate the above notion of plan. In particular, following (Levesque 1996), in the presence of boolean sensing actions we define a plan as a *conditional plan*, that is a program composed of action statements and if-then-else statements, such that each if-then-else statement occurs right after a sensing action statement, and is conditioned by the truth value of the sensed property. In other words, the structure of a plan corresponds to a binary tree, where each leaf corresponds to a state in which the goal holds: in our framework, such a tree is a subgraph of the graph corresponding to the FOE of the dynamic system.

Based on such a structure of plan, in the presence of sensing actions two different notions of plan have been defined in the literature:

- A *strong plan for G* is a conditional plan $\mathcal{S}_G$ such that the execution of $\mathcal{S}_G$ leads to a state in which G is known to hold *for each possible outcome of the sensing actions in S_G*.
- A *weak plan for G* is a conditional plan $\mathcal{S}_G$ such that *there exists an outcome of the sensing actions in S_G* for which the execution of $\mathcal{S}_G$ leads to a state in which G is known to hold.

A third notion of conditional plan, which can be considered as "intermediate" between the two above notions, extends the structure of plan by adding a "goto" statement, thus allowing cycles in the plan execution. In other words, under such an extended notion, plans correspond to graphs of a general form: in our framework, each such graph is a subgraph of the graph corresponding to the FOE of the dynamic system. We call *partially strong plan for G* a plan $\mathcal{S}_G$ such that, *for each possible outcome of the sensing actions in S_G*, if the execution of $\mathcal{S}_G$ terminates, then it leads to a state in which G is known to hold. This definition is equivalent to the definition of *strong cyclic plans* given in (Giunchiglia & Traverso 1999).

Namely, a partially strong plan is a plan that is not guaranteed to terminate: termination actually depends on the outcome of the sensing actions in the plan. However, if such a plan terminates, then it always leads to the goal. In this sense, a partially strong plan is both "weaker" than a strong plan (which both terminates and always reaches the goal) and "stronger" than a weak plan (which may terminate without reaching the goal).

Besides its theoretical interest, the notion of partially strong plan can be very useful in practice: for instance, if there exists no strong plan for G, then a partially strong plan constitutes the best reasonable way to attempt at reaching the goal, especially in the presence of an external mechanism that is able to cope with non-termination of the plan (the coordination module in our architecture), as illustrated in the next section.

Plan Representation

The representation of plans is given by *transition graphs*, where each node denotes an epistemic state, and is labeled with the properties that the robot knows to be true, and each arc denotes a state transition and is labeled with the action that causes the transition.

An epistemic state represents a set of world states (i.e. a set of situations the system can be in) and is characterized by a set of properties which give a partial description of the situation. Actions are represented using preconditions and effects. Preconditions are the conditions that are necessary for activating the action and indicate what must be true before the action is executed: they specify circumstances under which it is possible to execute an action. Effects are the conditions that must hold after the execution of the action and characterize how the state changes after the execution of the action.

The actions in the graph can be classified into ordinary (i.e. movement) actions and sensing actions. The former cause changes in the environment, while the latter permit the acquisition of information, in order to let the robot take better decisions. Both actions are relevant to state transition. Sensing actions are associated with conditions to be verified: depending upon the runtime value of these conditions, a different part of the plan will be executed.

Plan Generation

The plan generation module, given an initial state and a goal, generates a plan that, when executed starting from the initial state, if the execution terminates then the robot reaches a state in which the goal is satisfied.

The plan generation module selects a portion of the FOE of the KB containing only those actions that are necessary to achieve a goal starting from a given initial situation. In fact, conditional plans can in principle be generated in two steps (see in (Iocchi, Nardi, & Rosati 2000) for details). First, the FOE of the knowledge base is generated; this FOE can be seen as an action graph representing all possible plans starting from the initial state. Then, such a graph is visited, building a term (the cyclic conditional plan) representing a graph in which: (i) sensing actions generate branches; (ii) each branch leads either to a state satisfying the goal or to a previous state of the plan. However, the current implementation does not build the entire FOE before searching for the plan, but it builds the FOE starting from the initial state with a depth-first technique until a goal state is reached. In case of sensing actions all the branches are required to reach a goal state. In this way it is possible to extract a minimal plan (with respect to the number of actions to be executed).

If the planner finds a *strong plan*, this plan is represented as a tree in which sensing actions generate branches and each branch leads to a state satisfying the goal. However, in case a *strong plan* does not exists, the planner is able to build a *partially strong plan* that is not a tree anymore since there are also cycles. In the current implementation of our planner, for every sensing action that produces a portion of the model like the one shown in Fig. 2a, there are three possible cases: i) if neither path A nor B leads to the goal, then this part of the graph will be discarded; ii) if both path A and path B lead to a state in which the goal is verified, then this *strong plan* is returned; iii) if only one of the path A or B leads to the goal, while the other does not (Fig. 2b), then the planner is able to generate a *partially strong plan* like the one shown in Fig. 2c.

Implementation and experiments

The situation we describe in this section, as an example for explaining our plan generation and execution mechanism, is the dynamic exchange of the role of goalkeeper in the Sony Legged League and the application of the two-defender rule. The situation presented in Fig. 3a is a typical situation in the Sony Legged League matches in which the goalkeeper (robot number 1) is moving away from its own goal and is approaching the ball to push it away, while robot number 2 is far away from the ball and it cannot help the goalkeeper immediately. In this situation, it is more convenient for the team that robot 1 takes the role of attacker pushing the ball towards the opposite goal, while robot 2 goes back to defend its own goal acting as a goalkeeper. However, in performing this role exchange the two robots must comply with the two defenders rule, and thus robot 2 can enter the goal area only after robot 1 has left it.

This coordination problem cannot be solved by simply applying the distributed coordination protocol already presented in (Castelpietra *et al.* 2000), since dynamic role exchange is not sufficient to respect the two-defender rule. In other words, we want to solve two aspects of coordination: on the one hand, dynamic role exchange allows for selecting the best robot able to accomplish a task; on the other hand, while accomplishing their own tasks the robots must take into account some constraints on shared resources and thus have to synchronize their actions.

The problem of complying with the two-defender rule is solved by generating a plan in which one robot, before entering the goal area, must check that it is free (i.e. the other robot has left the area). This is achieved by adding in the knowledge base of the robot the specification of a sensing action *SenseFreeArea* that is used for verifying if the goal area is not occupied by any robot of the team.

The portion of the plan for the robot 2, of interest for this coordination problem, that has been generated by the plan generation module is shown in Fig. 3b. Notice that this is a *partially strong plan*, since it may be the case that its execution does not terminate. However, if the plan terminates, then the goal is achieved.

Plan execution

Plan execution is responsible for monitoring the actions that are executed by the robot and to perform state transitions when the actions terminate. During actual execution of the plan in real environments two issues must be faced: 1) failure of an action executed during the plan; 2) non-termination of the plan.

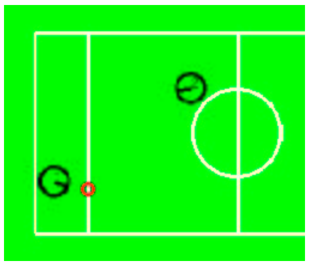
The first case is handled by the Plan Execution Module in this way: if, at the end of the execution of an action, the conditions in the world are different by the ones in the successor state (i.e. the effects of the action are not those described in the KB), then the Plan Execution Module will search for a *recovery state* in the current plan from which it is possible to continue the execution of the plan for reaching the goal. If this *recovery state* is not found, this module communicates a plan failure to an external module (in our case the Coordination Module) that will select another plan for the robot.

The second case is instead managed directly by the Coordination Module that guarantees that, if a robot is not able to perform a task, its role (and thus its plan) will be eventually changed (Castelpietra *et al.* 2000).

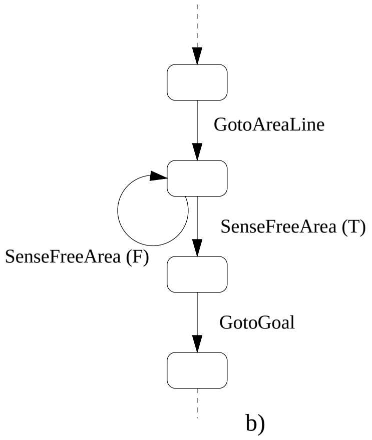
Experiments with a simulator

The experimental setting we have used has been given by a simulator (see Fig. 3a): even though it cannot provide a precise characterization of all the aspects that influence the performance of the robot in the real environment, it can provide useful feedback to the design of the coordination and plan execution modules for actual robots. Through this simulator we have verified the intended behaviour of the robots in each of the roles in different scenarios.

The simulator provides a global view of the environment that in the case of our experiment is the RoboCup soccer field, and shows ball position and robot positions and orientations. With the simulator it is possible to let the simulated robot play, changing the environment in order to represent some particular situations of interest. In our experiments



a)



b)

Figure 3: a) Typical situation for dynamic coordination. b) Plan generated for robot 2.

we have represented the situation of Figure 3, and we have checked the dynamic role assignment and the execution of the robot plans. More in depth, the first robot (robot number 1 in the figure) is initially assigned to a defensive role and its position is inside the goal area, robot number two starting position is outside the goal area. When the ball is positioned in front of the robot number 1, a role change occurs: robot number 1 takes the attack role and begins to push the ball towards the opponent goal escaping the goal area, meanwhile the robot number 2 takes the defender role, and starts going in the defense position but it does not enter the area until the other robot leaves it, due to the *SenseFreeArea* sensing action discussed above. The experiment shows that the algorithm for dynamic role assignment successfully exchange the role between the two robots and that the *SenseFreeArea* action correctly avoids the presence of two robots in the defense area.

The method used for the role exchange is also robust with respect to the following unexpected situation: a) robot 1 throws the ball towards robot 2 then, due to the dynamic role assignment algorithm, robot 2 becomes the attacker and begins to push the ball towards the opponent goal while robot 1 (which is become defender again) goes back to the defense position; b) if robot 1 is not able to escape from the goal area, then the utility function passes the role of attacker to robot 2; robot 2 starts heading towards the ball while robot 1 goes back to defense position; c) when three robots are playing, if robot 2 violates the two-defender rule and is taken out from the field (according to the RoboCup Legged competition rules) while robot 1 is pushing the ball, then robot 3 will take the defender role and will go in the defense position while robot 1 keeps on pushing the ball.

Conclusions

In this paper we have presented an extension of a previous approach to generate, based on a rich action theory, plans to be executed by our Cognitive Robots (Castelpietra *et al.* 2001).

The present work is focussed on *cyclic conditional plans*, for which partial correctness is guaranteed with respect to the formal specification of the system: if the plan terminates then the goal is achieved. More specifically, we have addressed the generation of plans containing a simple class of cycles that arise from the attempt to recover from the failure states originated by sensing actions. This kind of plans are useful for handling some aspects of coordination required in a multi-robot system, for instance the synchronization of actions during task execution.

As compared with previous work in Cognitive Robotics this is a novel attempt to generate plans that include cycles. As compared with the work on classical planning there is a close relationship with the work in (Bertoli *et al.* 2001), where only conditional plans (tree-structured) are generated. While we have considered a more general class of plans, we have not considered the efficient implementation of the planning algorithm, and we are currently addressing the application of model checking techniques, as done in (Bertoli *et al.* 2001), also in our setting. Moreover, we are extending our analysis to generate plans with cycles that are more general than the ones presented in this paper.

References

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

Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95).*

Castelpietra, C.; Iocchi, L.; Nardi, D.; Piaggio, M.; Scalzo, A.; and Sgorbissa, A. 2000. Communication and coordination among heterogeneous mid-size players: ART99. In *RoboCup-2000: Robot Soccer World Cup IV.*

Castelpietra, C.; Guidotti, A.; Iocchi, L.; Nardi, D.; and Rosati, R. 2001. Design and implementation of cognitive soccer robots. In *Proc. of RoboCup Symposium.*

Cimatti, A.; Giunchiglia, E.; Giunchiglia, F.; and Traverso, P. 1997. Planning via model checking: a decision procedure for $\mathcal{AR}$. In *Proc. of the 4th Eur. Conf. on Planning (ECP'97)*.

De Giacomo, G.; Iocchi, L.; Nardi, D.; and Rosati, R. 1996. Moving a robot: the KR&R approach at work. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*.

De Giacomo, G.; Iocchi, L.; Nardi, D.; and Rosati, R. 1997. Planning with sensing for a mobile robot. In *Proc. of 4th European Conference on Planning (ECP'97)*.

Donini, F. M.; Nardi, D.; and Rosati, R. 2002. Description logics of minimal knowledge and negation as failure. *ACM Trans. on Computational Logic* 3(2):1–49.

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

Iocchi, L.; Nardi, D.; and Rosati, R. 2000. 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)*, 678–689.

Iocchi, L. 1999. *Design and Development of Cognitive Robots*. Ph.D. Dissertation, Univ. "La Sapienza", Roma, Italy, On-line ftp.dis.uniroma1.it/pub/iocchi/.

Jensen, R. M.; Veloso, M. M.; and Bowling, M. H. 1999. OBDD-based optimistic and strong cyclic adversarial planning. In *Proc. of the 5th Eur. Conf. on Planning (ECP'99)*.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E.; and Matsubara, H. 1998. Robocup: A challenge problem for ai and robotics. In *Lecture Note in Artificial Intelligence*, volume 1395, 1–19.

Levesque, H. J. 1996. What is planning in presence of sensing? In Press, A. P. M., ed., *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, 1139–1149.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation, Special Issue on Action and Processes* 4(5):655–678.

Mc Cain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*.

Reiter, R. 2001. *Knowledge in action: Logical foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence* 89.