Planning with sensing, concurrency, and exogenous events: logical framework and implementation

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

Abstract

The focus of current research in cognitive robotics is both on the realization of systems based on known formal settings and on the extension of previous formal approaches to account for features that play a significant role for autonomous robots, but have not yet received an adequate treatment. In this paper we adopt a formal framework derived from Propositional Dynamic Logics by exploiting their formal correspondence with Description Logics, and present an extension of such a framework obtained by introducing both concurrency on primitive actions and autoepistemic operators for explicitly representing the robot's epistemic state. We show that the resulting formal setting allows for the representation of actions with context-dependent effects, sensing actions, and concurrent actions, and address both the presence of exogenous events and the characterization of the notion of executable plan in such a complex setting. Moreover, we present an implementation of this framework in a system which is capable of generating plans that are actually executed on mobile robots, and illustrate the experimentation of such a system in the design and implementation of soccer players for the 1999 Robocup competition.

1 INTRODUCTION

The focus of research in Cognitive Robotics is in devising actual implementations of reasoning about actions and action planning on top of mobile robots. To this purpose, several approaches have been proposed, based on various extensions of the Situation Calculus [Lesperance *et al.*,1998], Propositional Dynamic Logics (PDLs) [De Giacomo *et al.*,1996; 1997b], the Event Calculus [Shanahan,1997], and $\mathcal{A}$-languages and logic programming [Gelfond and Lifschitz,1993]. In pursuing this goal, two kinds of difficulties arise. On the one hand, the practical realization of formalisms on top of mobile robots requires the designer to put restrictions either on the expressiveness of the language or on the role played by automated reasoning. On the other hand, several features that have often been neglected by previous work on representation of dynamic systems and reasoning about actions are now receiving attention, because they are needed in practice: e.g., reasoning about the concurrent execution of actions [Lin and Shoham,1992; Baral and Gelfond,1993], sensing actions [Levesque,1996], and exogenous events [De Giacomo *et al.*,1997a].

In this paper we follow the approach to the representation of dynamic systems based on PDLs [Rosenschein,1981] and its extension based on the correspondence with Description Logics [De Giacomo *et al.*,1996; 1997b]. In fact, we extend such an approach in order to address action planning in the presence of sensing actions, concurrent actions and exogenous events, in the realm of Cognitive Robotics, namely with the goal of implementing it on top of a mobile robot.

An analysis of the state of the art in Cognitive Robotics shows that, while several semantic questions concerning the representation of complex dynamic domains have been addressed (see e.g. [Lin and Shoham,1992; Baral and Gelfond,1993; Reiter,1996; Pinto,1998] for the issue of representing concurrent actions), the problem of generating plans which can be actually executed by a robotic architecture is generally not considered. We thus propose a rich formal setting, where one can deal with sensing actions and concurrent execution of actions, and express static constraints and frame axioms with different flavors, still

retaining the ability of generating plans which can be actually executed by mobile robots. Moreover, we have experimented our approach in a real Cognitive Robotics application, namely the RoboCup99 F-2000 competition for soccer-playing robots, within the Azzurra Robot Team [Nardi *et al.*,1999].

The paper is organized as follows. In the next section we summarize the main features of the epistemic approach to representing dynamic systems. Then we discuss the representational features of our framework and, subsequently, reasoning. Finally, we report on the experimentation of our approach in the RoboCup competition.

2 AUTOEPISTEMIC APPROACH TO ACTION REPRESENTATION

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 logic $\mathcal{ALCK}_{\mathcal{NF}}$, and experimented its adequacy in the formalization of dynamic systems. Notably, 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 $\mathcal{A}$).

We start by pointing out the main capabilities that the logical framework has to provide, in order to fully support a suitable representation for the Cognitive Robotics applications under our examination.

Epistemic abilities First of all, we require a formalism in which it is possible to *explicitly* represent the robot's epistemic state. The reason for such a requirement is twofold:

1. an explicit representation of the robot's epistemic state allows for an easy characterization of effective plans, i.e. plans which can be actually executed by the robot (see Section 4);
2. as shown e.g. in [Scherl and Levesque,1993], an explicit representation of the robot's epistemic state allows for a natural formalization of 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.

Indeed, several epistemic approaches to reasoning about actions have been proposed in the literature, as well as the use of epistemic modalities for dealing with the representation of sensing actions (see e.g. [Scherl and Levesque,1993; Lakemeyer and Levesque,1998; Lobo *et al.*,1997]). However, such proposals do not address the issue of characterizing effective plans by exploiting the use of epistemic modal formalisms.

Concurrency We require the formalism to allow for reasoning about the concurrent execution of primitive actions, without introducing the notion of time to specify the start and the end of an action such as in [Pinto,1998; Reiter,1996] or adopting interleaving semantics [De Giacomo *et al.*,1997a]. We remark that we do not want to treat in an ad-hoc way each possible concurrent execution of actions, i.e., we do not define specific axioms for each possible combination of primitive actions: instead, we want to treat concurrency in a systematic way, deriving the possibility of concurrently executing two or more primitive actions directly from the specification of such actions.

Our formalization of concurrent actions is based on the following simple principle (for ease of exposition, below we illustrate the case of $n = 2$ concurrent actions: the case of an arbitrary n is obtained by a straightforward generalization).

Definition 1 *Two actions a_1, a_2 can be concurrently executed in a state s if and only if the following conditions hold: (i) both a_1 and a_2 can be executed in s; (ii) the effects of a_1 and a_2 are mutually consistent.*

Thus, as mentioned above, the possibility of concurrently executing two primitive actions only depends on the specification (in terms of preconditions and postconditions) of such primitive actions. In particular, the first condition imposes that a_1 and a_2 can be executed in s only if they can be individually executed in s, while the second condition prevents the concurrent execution of actions whose effects are not jointly consistent. We stress the fact that, while the first condition is independent of the current state s, the second condition actually depends on s, if, in the formalization, we allow context-dependent effects of actions (see next section) and/or inertia laws.

Other approaches have proposed the formalization of concurrency of primitive actions [Baral and Gelfond,1993; Giordano *et al.*,1998; Lin and Shoham,1992]. A novel feature of our approach is the possibility of modeling the concurrent execution of primitive actions and sensing actions.

Persistence and exogenous events We also require

the formalism to allow for the specification of suitable mechanisms (frame axioms) for expressing persistence of properties. In particular, we want to be able to formalize both *deterministic frame axioms*, stating that a certain property is guaranteed to persists (in the robot's epistemic state) after the execution of an action, and *default frame axioms*, stating that a property persists after the execution of an action if it is consistent with the effects of the action.

In this way, the formalism is able to represent both monotonic and nonmonotonic solutions to the frame problem, based on the automatic generation of frame axioms starting from the specification of the dynamic system (e.g. see [Reiter,1991]). However, we recall that such automatic solutions are effective only in simple dynamic settings satisfying severe restrictions on both the effects of actions and the static relationships between properties. Furthermore, 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. As we shall see in next section, this requires the possibility of formalizing non-inertial properties as well.

State and causal constraints Finally, we require the possibility of expressing both *state constraints* [Lin and Reiter,1994], that is, ordinary first-order sentences expressing static relationships between dynamic properties, and *epistemic constraints*, which can be seen as relationships which must be satisfied in each robot's epistemic state. In the following we will show that some forms of constraints that take into account *causal* dependencies between properties [Lin,1995; Mc Cain and Turner,1995; Thielscher,1997] can be considered as special forms of epistemic constraints, and will use such form of rules for specifying special kinds of frame axioms (epistemic frame axioms). Both ordinary state constraints and casual constraints enforce *ramifications*, e.g. indirect effects of actions.

Summarizing, to our knowledge none of the existing formal approaches to reasoning about actions in Cognitive Robotics is able to address *all* the above requirements for action representation.

The logic $\mathcal{ALCK}_{\mathcal{NF}}$ The formalism we employ for representing actions is basically the one presented in [De Giacomo *et al.*,1997b], extended with role conjunction in order to formalize the concurrent execution of primitive actions. Such a formalism is based on propositional dynamic logics (PDLs), and makes use of the tight correspondence between PDLs and description logics (DLs) [De Giacomo and Lenzerini,1994] that allows for considering PDLs and DLs as notational variants of each other. We use the notation of DLs, focusing on the well-known DL $\mathcal{ALC}$, corresponding to the standard PDL with atomic programs only. In addition, we use two nonmonotonic modal operators: a *minimal knowledge operator* $\mathbf{K}$ and a *default assumption operator* $\mathbf{A}$. These are interpreted according to the nonmonotonic modal logic *MKNF* [Lifschitz,1994], and give rise to the so-called autoepistemic description logic $\mathcal{ALCK}_{\mathcal{NF}}$ [Donini *et al.*,1997], which thus corresponds to a fragment of first-order *MKNF*.

Due to space limitations, we only briefly introduce $\mathcal{ALCK}_{\mathcal{NF}}$, and refer to [Donini *et al.*,1997] for a detailed introduction to such a formalism. The logic $\mathcal{ALCK}_{\mathcal{NF}}$ allows for representing a domain of interest in terms of *concepts* and *roles*. Concepts model classes of *individuals*, while roles model relationships between classes. Starting with atomic concepts and atomic roles, which are concepts and roles described simply by a name, complex concepts and roles can be built by means of the following abstract syntax:

$$\begin{array}{rcl} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \\ & & \exists R.C \mid \forall R.C \mid \mathbf{K}C \mid \mathbf{A}C \\ R & ::= & P \mid P_1 \sqcap \ldots \sqcap P_n \mid \mathbf{K}R \mid \mathbf{A}R \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) denotes a concept, P (possibly with a subscript) denotes an atomic role, and R denotes a role. An *$\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base* is the union of a set of inclusion assertions (called the TBox) and a set of instance assertions (called the ABox). Inclusion assertions have the form $C \sqsubseteq D$, where C, D are concept expressions. Instance assertions have the form $C(a)$ or $R(a, b)$, where C is a concept, R is a role, and a, b are *names* of individuals. We assume that different names denote different individuals.

As for the semantics (which is formally defined in the appendix), an interpretation structure in $\mathcal{ALCK}_{\mathcal{NF}}$ corresponds to a possible-world (Kripke-style) structure, in which each world corresponds to an interpretation of standard DLs. The interpretation structures of DLs are essentially graphs labeled both on nodes and edges. Nodes correspond to individuals: each node is labeled by concepts that denote the properties of the individual. Edges (called links in DL) are labeled by roles. To our purposes, it is important to point out how such interpretation structures can be put in correspondence with the states of a dynamic system (due to space limitations, we refer [De Giacomo *et al.*,1997b] for a more detailed description of this aspect).

Due to the form of the assertions used in the formalization of the dynamic system, $\mathcal{ALCK}_{\mathcal{NF}}$ models of a KB Σ can be interpreted as 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: 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 shown in [De Giacomo *et al.*,1997b], an epistemic inclusion assertion $\mathbf{K}C \sqsubseteq D$ can be naturally interpreted in $\mathcal{ALCK}_{\mathcal{NF}}$ in terms of a *rule*, i.e. a forward reasoning mechanism. For this reason, there is a strict semantic analogy between the sentence $\mathbf{K}C \sqsubseteq D$ and a *causal rule* [Mc Cain and Turner,1995] of the form $C \Rightarrow D$, and, in general, between epistemic constraints and causal relationships. Moreover, the default assumption operator $\mathbf{A}$ allows for expressing justifications of default rules, and the combined usage of $\mathbf{K}$ and $\mathbf{A}$ allows for formalizing defaults in terms of modal formulas [Lifschitz,1994]. In particular, the default rule $\frac{C(x):D(x)}{E(x)}$ (which can be interpreted as a special inference rule stating that, for each individual x, if $C(x)$ holds and it is consistent to assume $D(x)$, then $E(x)$ must be derived) can be represented in $\mathcal{ALCK}_{\mathcal{NF}}$ by means of the inclusion assertion $\mathbf{K}C \sqsubseteq \mathbf{A}\neg D \sqcup \mathbf{K}E$.

We can thus summarize the main features of $\mathcal{ALCK}_{\mathcal{NF}}$ for action representation:
(i) it allows for representing actions (through roles), states (through individuals), and state properties (through concepts);
(ii) it allows for explicitly representing the robot's knowledge, by means of two distinct autoepistemic modalities in the language;
(iii) it allows for a representation of sensing actions, through the use of the minimal knowledge operator;
(iv) it allows for the representation of concurrent actions, by allowing conjunction of binary roles;
(v) it allows for the representation of both default persistence of properties, through the use of the autoepistemic modalities, and non-inertial properties;
(vi) it allows for the representation of both ordinary state constraints and casual constraints.

3 REPRESENTING ACTIONS

In this section we present our formalization of actions in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework. Specifically, we represent a dynamic system in terms of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base $\Sigma = \Gamma_S \cup \Gamma_I \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR}$, where each Γ_x is defined below. In the following, we assume to deal with a set of primitive actions, partitioned into a set of *ordinary* actions, which are assumed to be deterministic actions with (possibly) context-dependent effects, and *sensing* actions, which are assumed to be actions able to sense boolean properties of the environment. We represent primitive actions in $\mathcal{ALCK}_{\mathcal{NF}}$ through a set of atomic role symbols, which we call *action-roles*.

State constraints

We call Γ_S the set of state constraints (also known as domain constraints) of our formalization. State constraints are used for representing background knowledge, which is invariant with respect to the execution of actions. We formalize state constraints as general $\mathcal{ALC}$ inclusion assertions, not involving action-roles, although in general they can involve other roles used for structuring concept (i.e. property) descriptions and form complex taxonomies of properties that are typical of DLs.

Initial state We denote as Γ_I the specification of the initial state in our formalization. Such a specification is given in terms of an instance assertion of the form $C(init)$, where C is an $\mathcal{ALC}$ concept, and *init* is the constant denoting the initial state. This axiom can be read as: C holds in the state *init* in every possible interpretation.

Precondition axioms for primitive actions We denote as Γ_P the set of precondition axioms which specify sufficient conditions for the execution of primitive actions in a state. Precondition axioms are expressed through autoepistemic sentences of the form:

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \tag{1}$$

where C is an $\mathcal{ALC}$ concept and R is a primitive action (either an ordinary or a sensing action). This axiom can be read as: if C holds in the current state s, then there exists a state s' which is the R-successor of s.

We remark the fact that the use of such form of precondition axioms assumes the possibility of identifying *sufficient* conditions for the execution of an action. Hence, this formalization is not able to deal with implicit qualifications: namely, if the execution of R in a state in which C holds gives rise to an inconsistent successor state (which can be caused by the interaction of action effects and state and epistemic constraints), then there is no interpretation structure satisfying the $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base, since the above precondition axiom enforces the existence of a successor state. Therefore, we conclude that the specification of the dynamic system is inconsistent.

Precondition axioms for concurrent actions In

order to formalize concurrent actions according to Definition 1, we introduce in our formalization of the dynamic system an autoepistemic axiom schema which states the preconditions under which two or more actions can be concurrently executed. In particular, we enforce the following axiom schema:

$$\exists \mathbf{K} R_1.\top \sqcap \exists \mathbf{K} R_2.\top \sqcap \neg \mathbf{A}(\forall R_1 \sqcap R_2.\bot) \sqsubseteq \exists \mathbf{K}(R_1 \sqcap R_2).\top \quad (2)$$

which encodes the preconditions for the concurrent execution of primitive actions R_1 and R_2 according to Definition 1. In fact, for each pair of actions R_1, R_2, and for each state s, R_1 and R_2 are concurrently executed in s if and only if both R_1 and R_2 are executed in s, and it is consistent to assume $\neg \forall R_1 \sqcap R_2.\bot$, i.e. the effects of R_1 and R_2 in s are mutually consistent. In the following, we implicitly assume that each $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base representing a dynamic system with primitive actions $R_1, \ldots, R_n$ contains all instances of the axiom schema $\exists \mathbf{K} R_{i_1}.\top \sqcap \ldots \sqcap \exists \mathbf{K} R_{i_k}.\top \sqcap \neg \mathbf{A}(\forall R_{i_1} \sqcap \ldots \sqcap R_{i_k}.\bot) \sqsubseteq \exists \mathbf{K}(R_{i_1} \sqcap \ldots \sqcap R_{i_k}).\top$, for $1 \leq k \leq n$, $1 \leq i_1 \leq \ldots \leq i_k \leq n$.

Notice that the above axiom schema is the only one needed in order to model concurrent actions in our formalization. In fact, due to the semantics of role conjunction, $R_1 \sqcap R_2(s, s')$ implies both $R_1(s, s')$ and $R_2(s, s')$, hence (due to the form of effect axioms) the effect of the concurrent execution of actions is modeled by the effect axioms relative to the primitive actions R_1 and R_2. For the same reason, persistence of properties during the concurrent execution of R_1 and R_2 is modeled by the frame axioms for R_1 and R_2.

Effect axioms Effect axioms specify the effects of executing an action in a given state. In order to specify the effects of actions, we distinguish between ordinary actions and sensing actions. First, ordinary effect axioms specify the effects of the execution of an ordinary action in a state satisfying given conditions. In our framework, we have that, if in the current state the property C holds, then, after the execution of the action R, the property D holds, in the following way:

$$\mathbf{K} C \sqsubseteq \forall R.D$$

Notice that, since we allow multiple effect axioms for the same action, we are able to formalize *context-dependent* actions, that is, the effect of an action may be different in two different states. Moreover, effect axioms for sensing axioms are of the form

$$\top \sqsubseteq \mathbf{K}(\forall R_S.D) \sqcup \mathbf{K}(\forall R_S.\neg D),$$

and specify the fact that, after the execution of the sensing action R_S, the robot knows whether the property D is true or false.

We call Γ_E the set of effect axioms in the formalization of the dynamic system.

Frame axioms Using the autoepistemic operators, it is possible to enforce different forms of inertia laws. In particular, we are able to specify *default frame axioms* i.e. default persistence rules which state that, if in the current state the property C holds, then, after the execution of the action R, the property C holds, if it is consistent with the effects of R. Due to the above mentioned possibility of expressing defaults in $\mathcal{ALCK}_{\mathcal{NF}}$, such a rule can be represented in our framework by the inclusion axiom

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} R.\mathbf{A} \neg C \sqcup \mathbf{K} C$$

and call Γ_{DFR} the set of default frame axioms in our specification. We also use *epistemic frame axioms* in our specification, which are able to express "causal" persistence rules. Such an axiom has the form

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} R.\neg \mathbf{K} D \sqcup \mathbf{K} C$$

and expresses the fact that property C is propagated only if property D holds in the successor state. We call Γ_{EFR} the set of epistemic frame axioms in Σ.

By suitably instantiating the above kinds of axioms in our system specification, we are able to formalize both inertial and non-inertial properties, and both inertial and non-inertial actions, thus addressing both the frame problem and the presence of exogenous events in our framework.

Notice that this kind of formalization correctly addresses the concurrent execution of a sensing action and an ordinary action, which is illustrated by the following example.

Example 2 Suppose the ordinary action R has the effect $\neg C$, while the sensing action S allows for knowing the truth value of the property D, and suppose both actions can be executed in those states in which the property C holds. Hence, Σ includes the precondition axioms $\mathbf{K} C \sqsubseteq \exists \mathbf{K} R.\top$ and $\mathbf{K} C \sqsubseteq \exists \mathbf{K} S.\top$, and the effect axioms $\top \sqsubseteq \forall R.\neg C$ and $\top \sqsubseteq (\mathbf{K} \forall S.D) \sqcup (\mathbf{K} \forall S.\neg D)$. Now suppose that all sensing actions are inertial: This can be formalized by adding to the specification a set of frame axioms of the form

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} S.\mathbf{A} \neg C \sqcup \mathbf{K} C \quad (3)$$

for each property C and for each sensing action S.

Suppose now that, in the current state (s) C is known by the robot: hence, both R (with successor state s_1)

and S (with successor state s_2) can be executed. Then, by Definition 1, the concurrent execution of R and S can be done in the current state (with successor state s_3). The effect of $R \sqcap S$ in s_3 is $\neg C \sqcap D$ (or $\neg C \sqcap \neg D$). Given our action formalization, the frame axiom (3) affects both s_2 and s_3, since both $\mathbf{K}S(s, s_2)$ and $\mathbf{K}S(s, s_3)$ hold. However, such an axiom causes the propagation of the property C only in the state s_2 and not in s_3, since in s_3 it is not consistent to assume that C holds, thus obtaining the desired behavior. □

Therefore, the $\mathcal{ALCK}_{\mathcal{NF}}$-based framework for action formalization matches the representational requirements illustrated in Section 2. Hence, it constitutes the first formal framework that allows for representing *all* the required features of a dynamic system, and provides for reasoning methods applicable to plan generation, as shown in the next section.

4 PLANNING IN $\mathcal{ALCK}_{\mathcal{NF}}$

In this section we study planning in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework and present a reasoning method for generating both conditional and concurrent plans.

The planning problem The classical formulation of the notion of plan existence corresponds to the following: given the specification of the dynamic system and of the initial state, and a goal (property) G, there exists a plan for G if and only if there 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.

1. We require that the robot is aware of the fact that the goal is reached, i.e. we want the robot to know that G holds after the execution of $\mathcal{S}$.
2. We require that the plan can be effectively executed by the robot: this implies that, for each i, if s_{i+1} is the state resulting from the execution of the i-th action of the sequence $\mathcal{S}$ in state s_i, then the robot knows that the action can be executed in s_i (i.e. it knows that the precondition of the action holds in state s_i).
3. 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 planning problem. In particular, following [Levesque,1996], in 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. A 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.
4. The possibility of having concurrent executions of actions further extends the above notion of conditional plan to programs containing statements denoting the concurrent execution of primitive actions.

It can be shown that the above notion of plan existence can be reduced in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework to a reasoning problem:

$$\Sigma \models C_G(init) \tag{4}$$

where C_G is *any* concept belonging to the set $\mathcal{P}_G$ defined inductively as follows: (i) $\mathbf{K}G \in \mathcal{P}_{\mathcal{C}}$; (ii) if $C_1, \ldots, C_m \in \mathcal{P}_C$, then $\exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R_{S_1} \parallel \ldots \parallel R_{S_l})$.$(\mathbf{K}S_1 \sqcap \ldots \sqcap \mathbf{K}S_l \sqcap C_1) \sqcup \ldots \sqcup (\mathbf{K}\neg S_1 \sqcap \ldots \sqcap \mathbf{K}\neg S_l \sqcap C_m) \in \mathcal{P}_{\mathcal{C}}$ for each $0 \leq k, l \leq n$, where $m = 2^{k+l}$, each R_{M_i} is an ordinary action and each R_{S_i} is a sensing action for the property S_i.

Remark. The above characterization intentionally does not take into account *implicit* (or indirect) forms of nondeterminism in the execution of an action in a given state. In fact, due to the presence of both state (and epistemic) constraints and default frame axioms, even primitive, ordinary actions may give rise in general to multiple possible epistemic states. Such a form of nondeterminism is not handled by our notion of effective plan, and we make the assumption that the specification Σ does not give rise to implicit nondeterminism.[1] Indeed, it may be possible that a conditional plan, dealing with this form of nondeterminism, exists in such cases. However, in the present work we are not interested in these situations, since we want to enforce non-sensing actions to be deterministic (although context-dependent) actions.

Reasoning method To the aim of generating conditional plans with concurrent execution of actions in the proposed framework, we extend the algorithm described in [De Giacomo *et al.*,1997b] for computing the *first-order extension* (FOE) of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge

[1] It can be effectively checked whether Σ satisfies such a property, by verifying, in the algorithm reported in Figure 1, whether there exist two (or more) different default propagations of properties in the generation of the successor state.

base $\Sigma = \Gamma_S \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR} \cup \Gamma_I$ containing the formalization of the dynamic system in the terms described above. Informally, given a specification Σ and a goal G, $FOE(\Sigma, G)$ is an $\mathcal{ALC}$ knowledge base which consists of: (1) the static axioms in Γ_S; (2) the specification of the initial state (the assertions on *init* in Γ_I) augmented by the assertions which are consequences (up to renaming of individuals) of the epistemic sentences in Σ. The FOE of Σ provides a unique characterization of the knowledge that is shared by all the models of Σ, which is relevant with respect to the planning problem for the goal G.

The FOE of Σ for G is computed by the algorithm shown in Fig. 1. Roughly speaking, the algorithm, starting from the initial state *init*, applies to each state which does not satisfy the goal G the rules in the sets Γ_E, Γ_{DFR}, and Γ_{EFR} which are triggered by such a state, generating a new successor state, unless one with the same properties has already been created. The effects of the epistemic axioms are thus computed, and a non-epistemic "completion" of the knowledge base is obtained. In the algorithm,

$$\text{CONCEPTS}(\Gamma_S \cup \mathcal{A}, s) = \{C \mid \Gamma_S \cup \mathcal{A} \models C(s)\}$$

denotes the set of concepts that are *valid* for the explicitly named individual s, occurring in the set of instance assertions $\mathcal{A}$, wrt the $\mathcal{ALC}$ knowledge base $\Gamma_S \cup \mathcal{A}$, and $\text{POST}(s, R, \Gamma_S \cup \mathcal{A}, \Gamma_x) = \{D \mid \mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{K}D \in \Gamma_x \text{ and } \Gamma_S \cup \mathcal{A} \models C(s)\}$ denotes the effect of the application of all the triggered rules belonging to the set Γ_x involving the action R in the state s, namely the set of postconditions (concepts) of the rules in Γ_x which are triggered by s.

With respect to the algorithm described in [De Giacomo *et al.*,1997b] the following new issues have to be taken into account: (i) adding new transitions representing concurrent actions; (ii) verifying consistency of concurrent action execution; (iii) applying the new form of frame axioms. Hence the algorithm, for each subset of the set of possible actions in a given state s, attempts to create a new successor state with respect to the concurrent action. To this end, effect axioms are first applied for each single action, then consistency of the union of the sets of axioms obtained for each single action is verified.

It can be shown that the FOE is unique, that is, every order of extraction of states from the set ACTIVE_STATES and of the frame axioms from Γ_{DFR} and Γ_{EFR} produces the same set of assertions,[2] up to renaming of states. Moreover, the condition

[2]This property is a consequence of the assumption that Σ does not allow for implicit nondeterminism.

ALGORITHM FOE
INPUT: $\Sigma = \Gamma_S \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR} \cup \Gamma_I$, goal G
OUTPUT: $FOE(\Sigma, G)$ if Σ is consistent,
ERROR otherwise

PROCEDURE CreateNewState($s, \{R_1, \ldots, R_k\}$)
begin
s' = NEW state name;
$\mathcal{A}' = \mathcal{A} \cup \{R_1 \sqcap \ldots \sqcap R_k(s, s')\} \cup$
$\{D(s') \mid 1 \leq i \leq k \wedge D \in \text{POST}(s, R_i, \Gamma_S \cup \mathcal{A}, \Gamma_E)\}$;
if $\Gamma_S \cup \mathcal{A}' \not\models \bot$
then begin
for each axiom $\mathbf{K}C \sqsubseteq \forall \mathbf{K}R_i.\mathbf{A}\neg C \sqcup \mathbf{K}C \in \Gamma_{DFR}$
such that $1 \leq i \leq k \wedge \Gamma_S \cup \mathcal{A} \models C(s)$ **do**
if $\Gamma_S \cup \mathcal{A}' \not\models \neg C(s')$ **then** $\mathcal{A}' = \mathcal{A}' \cup \{C(s')\}$;
for each axiom $\mathbf{K}C \sqsubseteq \forall \mathbf{K}R_i.\neg \mathbf{K}D \sqcup \mathbf{K}C \in \Gamma_{EFR}$
such that $1 \leq i \leq k \wedge \Gamma_S \cup \mathcal{A} \models C(s)$ **do**
if $\Gamma_S \cup \mathcal{A}' \models D(s')$ **then** $\mathcal{A}' = \mathcal{A}' \cup \{C(s')\}$;
if there exists a state $s'' \in$ ALL_STATES
such that
CONCEPTS($\Gamma_S \cup \mathcal{A}, s''$)=CONCEPTS($\Gamma_S \cup \mathcal{A}', s'$)
then $\mathcal{A} = \mathcal{A} \cup \{R_1 \sqcap \ldots \sqcap R_k(s, s'')\}$
else begin
$\mathcal{A} = \mathcal{A}'$;
ACTIVE_STATES = ACTIVE_STATES $\cup\{s'\}$;
ALL_STATES = ALL_STATES $\cup\{s'\}$
end
end
else if $k = 1$ **then** EXIT WITH ERROR;
end;

begin
$\mathcal{A} = \Gamma_I$; ACTIVE_STATES = $\{init\}$;
ALL_STATES = $\{init\}$;
repeat
s = choose(ACTIVE_STATES); EXEC = $\emptyset$;
if $\Gamma_S \cup \mathcal{A} \not\models G(s)$
then begin
for each ordinary action R_M **do**
if there exists $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R_M.\top \in \Gamma_P$
such that $\Gamma_S \cup \mathcal{A} \models C(s)$
then begin
CreateNewState($s, \{R_M\}$);
EXEC = EXEC $\cup\{R_M\}$
end;
for each sensing action R_S **do**
if there exists $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R_S.\top \in \Gamma_P$
such that $\Gamma_S \cup \mathcal{A} \models C(s)$
and $\Gamma_S \cup \mathcal{A} \not\models S(s)$
and $\Gamma_S \cup \mathcal{A} \not\models \neg S(s)$
then begin
CreateNewState($s, \{R_S^+\}$);
CreateNewState($s, \{R_S^-\}$);
EXEC = EXEC $\cup\{R_S^+, R_S^-\}$
end;
for each subset $R_1, \ldots, R_k$ ($k \geq 2$) of EXEC **do**
CreateNewState($s, \{R_1, \ldots, R_k\}$);
ACTIVE_STATES = ACTIVE_STATES $-\{s\}$
end
until ACTIVE_STATES = $\emptyset$;
return $\Gamma_S \cup \mathcal{A}$
end.

Figure 1: Algorithm computing $FOE(\Sigma)$

CONCEPTS($\Gamma_S \cup \mathcal{A}, s$) =CONCEPTS($\Gamma_S \cup \mathcal{A}', s'$) can be checked by verifying whether, for each concept C such that either $C(init) \in \Gamma_I$ or C is in the postcondition of some axiom in $\Gamma_E \cup \Gamma_{EFR} \cup \Gamma_{DFR}$, $\Gamma_S \cup \mathcal{A} \models C(s)$ iff $\Gamma_S \cup \mathcal{A}' \models C(s')$. Finally, it is easy to see that the total number of different instances of the above sets of concepts is finite (exponential in the number of axioms in Σ). Hence, the condition ACTIVE_STATES= $\emptyset$ is eventually reached, which implies that the algorithm terminates.

The notion of first-order extension constitutes the basis of a sound and complete planning method for the $\mathcal{ALCK}_{\mathcal{NF}}$ framework. More specifically, we show that the planning problem in Σ expressed by (4) can be reduced to an entailment problem for $FOE(\Sigma, G)$, by making use of the following translation function $\tau(\cdot)$.

Definition 3 *Let C be a concept expression representing a plan (i.e. belonging to the set $\mathcal{P}_\mathcal{C}$). Then, $\tau(C)$ is the concept expression obtained as follows:*
(i) if $C = \mathbf{K}G$ then $\tau(C) = \mathbf{K}G$;
(ii) if $C = \exists \mathbf{K} R_{M_i}.C_1$ then $\tau(C) = \exists \mathbf{K} R_{M_i}.\tau(C_1)$;
(iii) if $C = \exists \mathbf{K} R_{S_i}.(\mathbf{K}S_i \sqcap C_1) \sqcup (\mathbf{K}\neg S_i \sqcap C_2)$ then $\tau(C) = \exists \mathbf{K} R^+_{S_i}.\tau(C_1) \sqcap \exists \mathbf{K} R^-_{S_i}.\tau(C_2)$;
(iv) if $C = \exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R_{S_1} \parallel \ldots \parallel R_{S_l}).(\mathbf{K}S_{i_1} \sqcap \ldots \sqcap \mathbf{K}S_{i_j} \sqcap \mathbf{K}\neg S_{i_{j+1}} \sqcap \ldots \sqcap \mathbf{K}\neg \mathcal{S}_{i_l} \sqcap C_1)$ then $\tau(C) = \exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R^+_{S_{i_1}} \parallel \ldots \parallel R^+_{S_{i_j}} \parallel R^-_{S_{i_{j+1}}} \parallel \ldots \parallel R^-_{S_{i_l}}).\tau(C_1)$.

Theorem 4 *Let $C \in \mathcal{P}_\mathcal{C}$. Then, $\Sigma \models C(init)$ iff $FOE(\Sigma, G) \models \tau(C)(init)$.*

In this way the planning problem consists of generating a constructive proof of the logical entailment $FOE(\Sigma, G) \models \Pi_G(init)$, where Π_G is a concept expression representing a plan for achieving the goal G from the initial situation *init*.

Planner implementation The framework and the algorithm described in the previous sections have been actually implemented and used to describe the knowledge of actual mobile robots embedded in real environments. The current implementation of the planner is built on top of the reasoning services provided by the well-known DL system CLASSIC [Borgida *et al.*,1989]. In particular, we make use of its built-in instance checking mechanism to check the validity of a concept in a state, and of triggering of rules to propagate effects. However, CLASSIC does not provide for a full implementation of the formalism. In particular, the epistemic operators **K** and **A** are handled by ad hoc attached procedures, the union operator is allowed only in the preconditions of the axioms, and negation is dealt with by the use of additional concepts.

The planning procedure, given an initial state and a goal, generates a conditional plan with concurrent actions that, when executed starting from the initial state, leads to a state in which the goal is satisfied. This procedure is based on the result of Theorem 4 and, specifically, aims at building a term (concept) Π_G such that the implication $FOE(\Sigma) \models \Pi_G(init)$ holds.

Such a plan could in principle be generated in two steps. First, the FOE of the knowledge base is generated (using the algorithm above): such a knowledge base 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 plan) representing a tree in which: (i) sensing actions generate branches; (ii) concurrent actions are explicitly represented in the arcs; (iii) each branch leads to a state satisfying the goal. Obviously, several strategies can be applied to implement this method, and they are not addressed in this paper.

5 APPLICATION DOMAINS

In this section we present some examples of application of the above formalism, addressing its most relevant features. On the one hand, we refer to classical examples such as the vase-on-the-table [Lin and Shoham,1992; Gelfond and Lifschitz,1993; Giordano *et al.*,1998]. The formalization of this example, which is omitted here for lack of space, shows that, in our framework, the correct plan can be obtained by combining the use of epistemic and state constraints with the possibility of executing concurrent actions. On the other hand, we discuss a concrete application domain. The framework and the planning system presented above have been used to describe the knowledge of actual mobile robots embedded in dynamic office-like environments (see [Iocchi,1999] for a detailed description of the integration of the framework within the mobile robots) and in the scenario of robotic soccer players provided by RoboCup competitions. In the implementation, our planner generates Colbert activities (control programs) that can be directly executed by the Saphira system [Konolige *et al.*,1997], which controls the mobile base.

Before we introduce the examples, it is worth pointing out that our aim here is not to describe the full implementation, but rather to focus on the generation of plans that highlight the features of the formalism. In particular, we do not address here the execution mechanism and the problem of splitting the tasks to be accomplished between the reasoning component and the underlying control level. By exploiting the proposed approach, we have been able to formal-

$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq$
$\quad \mathbf{K}(BallClose \sqcap OpponentOnBall) \sqsubseteq \exists \mathbf{K} tackle.\top$
$\mathbf{K}(\neg BallClose \sqcap OpponentOnBall) \exists \mathbf{K} intercept.\top$
$\mathbf{K}(BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} kick.\top$
$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} goToBall.\top$

$\mathbf{K}\top \sqsubseteq \forall tackle.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall intercept.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall kick.(GoalProtected \sqcap \neg BallClose)$
$\mathbf{K}\top \sqsubseteq \forall goToBall.(GoalProtected \sqcap BallClose)$

$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} senseBallClose.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseBallClose.BallClose) \sqcup$
$\quad \mathbf{K}(\forall senseBallClose.\neg BallClose)$
$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} opponentOnBall.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseOpponentOnBall.OpponentOnBall) \sqcup$
$\quad \mathbf{K}(\forall senseOpponentOnBall.\neg OpponentOnBall)$

Figure 2: $\mathcal{ALCK}_{\mathcal{NF}}$-KB for Example 1.

ize at the logical level several situations arising in the RoboCup scenario and to generate, through the planner, a significant portion of the control programs that were executed on some of the soccer players that participated, within the ART team, in the RoboCup-99 F-2000 league [Nardi *et al.*,1999].

Example 1: Planning defensive actions In the first example we consider the case of two actions which sense fluents that can change by the effect of exogenous events (i.e., non-persistent fluents), and, therefore, may need to be executed concurrently.

Suppose that one wants to represent a defensive situation, where the goal of the robot is to protect its own goal. The robot can see the ball $BallInLPS$, and there are two sensing actions that allow to decide whether the ball is close ($BallClose$) and whether there is an opponent which is on the ball performing some offensive action ($OpponentOnBall$). Moreover, there are various actions that the robot can take to properly defend its goal: *tackle*, which is required when both robots are next to the ball; *intercept*, (i.e. put itself between the opponent and the goal) which must be preferred when the opponent is on the ball and the robot is not; *kick*, which can be accomplished when the robot is close to the ball and the opponent is not; $goToBall$, which should be chosen when the opponent is not on the ball. It is worth mentioning that there are real differences in the effectiveness of the above actions depending on the situation. For example when the ball is in between the robot and the opponent kicking (forward) is usually not very effective and may lead to a foul and a tackling action is more appropriate.

The knowledge base reported in Figure 2 reports precondition and effect axioms for moving and sensing actions. Given the initial situation $BallInLPS$ and the goal $GoalProtected$, the system generates the following conditional plan:

```
senseBallClose || senseOpponentOnBall;
if (BallClose and not OpponentOnBall)
  { kick; }
  else { if (BallClose and OpponentOnBall)
       { tackle; }
       else { if (not BallClose and OpponentOnBall)
            { intercept; }
            else { goToBall; }
```

Notice that if the two sensing actions are not executed concurrently, there is no plan, because neither $BallClose$ nor $OpponentOnBall$ persists after the execution of a sensing action. Furthermore, the goal is achieved whatever is the result of the sensing actions. It is worth emphasizing that, in general, the condition that the plan always leads to satisfy the goal is very strong. In fact, it is often the case that only certain paths in the action graph generated by computing the FOE lead to a state satisfying the goal. In such cases one can still extract from the FOE indications to drive the actions of the robot, but we cannot further exploit this issue here. Moreover, in practice, the execution of plans may fail due to the uncertainty and dynamics of the domain. The verification of such failures is embedded into the execution mechanism, whose details again cannot be provided here.

Example 2: Planning the pass In the RoboCup scenario, coordination among the robots is a critical issue, and we want to represent the robot's knowledge that allows for reasoning about concurrent actions performed by two or more robots. In particular, we shall refer to one of the challenges that have been proposed for coordinating players in RoboCup, namely the pass. We consider a situation in which two players $P1$ and $P2$ are executing an offensive action towards the opponent goal. Initially, $P1$ has possession of the ball ($BallPoss_1$) and can freely move ahead towards the opponent goal ($FreeAhead_1$). The idea is that $P1$ can move forward till it reaches a good position to shoot. At this point, it can check whether there is still the possibility to kick forward ($FreeAhead_1$ still holds) or else decide to pass to $P2$ to have a better chance to score. From this specification, it follows that each robot must be able to perform the following actions: *fwdKeepingBall* to move forward trying to keep ball possession, *positionForPass* to get to a proper position to receive a pass and conclude the action, *kick* for kicking the ball, *pass* to pass the ball, *receiveAndKick* to get the pass and conclude the action, and finally *senseFreeAhead* for detecting whether there are obstacles between itself and the opponent goal. We model

$\mathbf{K}(BallPoss_i \sqcap \neg FreeAhead_i \sqcap ShootPsn_j) \sqsubseteq$
$\quad \mathbf{K}(BallPoss_i \sqcap FreeAhead_i) \sqsubseteq \exists \mathbf{K} fwdKeepingBall_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_i \sqcap FreeAhead_i) \sqsubseteq \exists \mathbf{K} kick_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_j \sqcap \neg FreeAhead_i) \sqsubseteq \exists \mathbf{K} pass_i^j.\top$
$\mathbf{K}(BallClose_j \sqcap ShootPsn_j) \sqsubseteq \exists \mathbf{K} receiveAndKick_j.\top$
$\mathbf{K} BallPoss_i \sqsubseteq \exists \mathbf{K} positionForPass_j.\top$

$\mathbf{K}\top \sqsubseteq \forall fwdKeepingBall_i.(BallPoss_i \sqcap ShootPsn_i)$
$\mathbf{K}\top \sqsubseteq \forall kick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall pass_i^j.BallClose_j$
$\mathbf{K}\top \sqsubseteq \forall receiveAndKick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall positionForPass_i.ShootPsn_i$

$\mathbf{K}\top \sqsubseteq \exists \mathbf{K} senseFreeAhead_i.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseFreeAhead_i.FreeAhead_i) \sqcup$
$\quad \mathbf{K}(\forall senseFreeAhead_i.\neg FreeAhead)$

$\mathbf{K} ShootPsn_i \sqsubseteq \forall \mathbf{K} pass_j^i.\mathbf{A} \neg ShootPsn_i \sqcup \mathbf{K} ShootPsn_i$
...

Figure 3: $\mathcal{ALCK}_{\mathcal{NF}}$-KB for Example 2.

this scenario through the $\mathcal{ALCK}_{\mathcal{NF}}$-KB reported in Figure 3, in which the subscript $i, j \in \{1, 2\} \wedge i \neq j$ denotes actions and conditions concerning each robot.

Notice that the fluent representing the possession of the ball ($BallPoss$) does not persist, and, unless explicitly specified, as in the case of $fwdKeepingBall$, it must be verified by a sensing action; instead, $ShootPsn$ (representing the property of being in a good position for shooting) persists when a pass is executed, and persists by default during the execution of other actions.

Given the initial state in which $BallPoss_1 \sqcap FreeAhead_1$ holds and the goal $BallKicked$, we obtain the following plan.

```
senseFreeAhead1 || fwdKeepingBall1 || positionForPass2;
if (FreeAhead1)
  { kick1 }
  else { pass12; receiveAndKick2;}
```

We remark that the two actions $fwdKeepingBall_1$ and $senseFreeAhead_1$ must be executed concurrently, because $FreeAhead$ does not persist; moreover, the previous actions must be executed concurrently with $positionForPass_2$, because in any of the two corresponding sequences of moves of the two players, there is the possibility that an opponent moves the ball while $P1$ is waiting for the positioning of $P2$.

Plan execution. The introduction of a system that generates plans with concurrent actions requires the robotic architecture to be able to schedule and manage concurrent behaviors and to provide synchronization among such behaviors. Concurrent plans are actually executed on a single player by making use of the Saphira built-in mechnisms for activating concurrent behaviors and for monitoring their end before starting the next action in the plan. The execution of global concurrent plans (that concern more than one robot) is instead realized by means of explicit communication among players. Observe that in this case all the players share the same plan, and each player is able to identify the actions that must be executed. For example, the execution of the action $A_1||B_2$ is obtained by performing A on $P1$ and B on $P2$ and by a broadcast notification when actions terminate. In this way, all the robots involved in the global plan can detect when it is possible to start the next action in the plan.

6 CONCLUSIONS

In this paper we have presented a new framework for representing actions and generating plans, that can be executed by a (cognitive) mobile robot, from a declarative specification of the knowledge about the dynamic system. We believe that the most relevant aspects of the work presented in the paper are essentially two. The first one is the expressiveness of the proposed framework, which allows for representing epistemic states of the agent, sensing actions, primitive concurrent actions, state constraints. Many such features had been considered by previous work, however their combination gives rise to a new effective characterization of context-dependent actions and exogenous events. In addition, we have addressed plan generation in such a rich framework, which is often not considered by previous work, focusing mainly on the epistemological aspects of the formalization.

The second aspect worth remarking is that the framework has been implemented and used to generate plans for a concrete multi-robot scenario, provided by the RoboCup mid-size competition for robotic soccer. The examples presented should provide enough evidence that the representational features of the formalism are necessary in order to correctly model such a domain.

Several issues arising from the work presented in the paper could be investigated following the proposed approach. One interesting question, we believe, is how to embed the plans obtained from the specification into the execution mechanism. More precisely, the programs executed on the robots take the form of a transition graph. Even though we can currently generate portions of such a graph from the extracted plans, there are aspects of the actual execution, such as monitoring the failure and the termination of actions, that we deal with by ad-hoc ways, while a more systematic treatment would be needed.

References

[Baral and Gelfond, 1993] C Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI'93)*, pages 866–871, 1993.

[Borgida *et al.*, 1989] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[De Giacomo and Lenzerini, 1994] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[De Giacomo *et al.*, 1996] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. 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)*, 1996.

[De Giacomo *et al.*, 1997a] G. De Giacomo, H. J. Levesque, and Y. Lesperance. Reasoning about concurrent executions, prioritized interrupts, and exogenous action in the situation calculus. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, 1997.

[De Giacomo *et al.*, 1997b] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing for a mobile robot. In *Proc. of the Fourth European Conf. on Planning (ECP'97)*, number 1348 in Lecture Notes In Artificial Intelligence, pages 156–168. Springer-Verlag, 1997.

[Donini *et al.*, 1997] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Autoepistemic description logics. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, pages 136–141, 1997.

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

[Giordano *et al.*, 1998] L. Giordano, A. Martelli, and C. Schwind. Dealing with concurrent actions in modal action logic. In *Proc. Of European Conference on Artificial Intelligence (ECAI'98)*, 1998.

[Iocchi, 1999] Luca Iocchi. *Design and Development of Cognitive Robots*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1999.

[Konolige *et al.*, 1997] K. Konolige, K.L. Myers, E.H. Ruspini, and A. Saffiotti. The Saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):215–235, 1997.

[Lakemeyer and Levesque, 1998] Gerhard Lakemeyer and Hector J. Levesque. AOL: a logic of acting, sensing, knowing, and only knowing. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 316–327. Morgan Kaufmann, Los Altos, 1998.

[Lesperance *et al.*, 1998] Yves Lesperance, Kenneth Tam, and Michael Jenkin. Reactivity in a logic-based robot programming framework. In *Proceedings of AAAI98 Fall Symposium on Cognitive Robotics*, 1998.

[Levesque, 1996] 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.

[Lifschitz, 1994] Vladimir Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[Lin and Reiter, 1994] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation*, 4(5):655–678, 1994.

[Lin and Shoham, 1992] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 590–595, 1992.

[Lin, 1995] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, 1995.

[Lobo *et al.*, 1997] Jorge Lobo, Gisela Mendez, and Stuart R. Taylor. Adding knowledge to the action description language A. In *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI'97)*, pages 454–459, 1997.

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

[Nardi *et al.*, 1999] Daniele Nardi, Giovanni Adorni, Andrea Bonarini, Riccardo Cassinis, Giorgio Clemente, Antonio Chella, Enrico Pagello, and Maurizio Piaggio. ART: Azzurra Robot Team. Available at `http://www.ida.liu.se/ext/RoboCup-99`, 1999.

[Pinto, 1998] Javier A. Pinto. Concurrent actions and interacting effects. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Reiter, 1996] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 2–13, 1996.

[Rosenschein, 1981] Stanley J. Rosenschein. Plan synthesis: A logical perspective. In *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence (IJCAI'81)*, pages 331–337, 1981.

[Scherl and Levesque, 1993] 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.

[Shanahan, 1997] M. Shanahan. Event calculus planning revisited. In *Proc. of the 4th European Conference on Planning (ECP'97)*, 1997.

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

Appendix: $\mathcal{ALCK}_{\mathcal{NF}}$ semantics

Since the logic $\mathcal{ALCK}_{\mathcal{NF}}$ is a particular case of Lifschitz's logic *MKNF* [Lifschitz,1994], the semantics of $\mathcal{ALCK}_{\mathcal{NF}}$ is obtained by interpreting concepts and roles on possible-world structures corresponding to sets of first-order interpretations. Non-epistemic concepts are interpreted as subsets of a domain, while non-epistemic roles are interpreted as binary relations over such a domain. Formally, an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{V}(\mathcal{I})})$ consists of a *domain of interpretation* Δ, and an *interpretation function* $\cdot^{\mathcal{V}(\mathcal{I})}$ mapping each concept to a subset of Δ (we assume $\top^{\mathcal{V}(\mathcal{I})} = \Delta$ and $\bot^{\mathcal{V}(\mathcal{I})} = \emptyset$) and each role to a subset of $\Delta \times \Delta$ as follows (A denotes an atomic concept, C, possibly with a subscript, denotes a concept, P, possibly with a subscript, denotes an atomic role, and R denotes a role):

$$
\begin{array}{rcl}
A^{\mathcal{V}(\mathcal{I})} & \subseteq & \Delta \\
(\neg C)^{\mathcal{V}(\mathcal{I})} & = & \Delta \setminus C^{\mathcal{V}(\mathcal{I})} \\
(C_1 \sqcap C_2)^{\mathcal{V}(\mathcal{I})} & = & C_1^{\mathcal{V}(\mathcal{I})} \cap C_2^{\mathcal{V}(\mathcal{I})} \\
(C_1 \sqcup C_2)^{\mathcal{V}(\mathcal{I})} & = & C_1^{\mathcal{V}(\mathcal{I})} \cup C_2^{\mathcal{V}(\mathcal{I})} \\
(\exists R.C)^{\mathcal{V}(\mathcal{I})} & = & \{d \in \Delta \mid \exists d'.(d,d') \in R^{\mathcal{V}(\mathcal{I})} \\
 & & \text{and } d' \in C^{\mathcal{V}(\mathcal{I})}\} \\
(\forall R.C)^{\mathcal{V}(\mathcal{I})} & = & \{d \in \Delta \mid \forall d'.\ \text{if } (d,d') \in R^{\mathcal{V}(\mathcal{I})} \\
 & & \text{then } d' \in C^{\mathcal{V}(\mathcal{I})}\} \\
P^{\mathcal{V}(\mathcal{I})} & \subseteq & \Delta \times \Delta \\
(P_1 \sqcap \ldots \sqcap P_n)^{\mathcal{V}(\mathcal{I})} & = & (P_1)^{\mathcal{V}(\mathcal{I})} \cap \ldots \cap (P_n)^{\mathcal{V}(\mathcal{I})}
\end{array}
$$

The semantics of epistemic sentences is based on the following *Common Domain Assumption*: in each possible-world structure, each interpretation is defined over the same, fixed, countable-infinite domain of individuals Δ. We define an *epistemic interpretation* as a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ where $\mathcal{I}$ is an interpretation and $\mathcal{M}, \mathcal{N}$ are sets of interpretations such that $\mathcal{I} = (\Delta, \cdot^{\mathcal{V}(\mathcal{I})})$ and all interpretations in $\mathcal{M}$ and $\mathcal{N}$ are defined over the domain Δ. Epistemic sentences are interpreted on epistemic interpretations as follows (again, we assume that $(\top)^{\mathcal{I},\mathcal{M},\mathcal{N}} = \Delta$ and $(\bot)^{\mathcal{I},\mathcal{M},\mathcal{N}} = \emptyset$):

$$
\begin{array}{rcl}
(A)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & A^{\mathcal{V}(\mathcal{I})} \\
(\neg C)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \Delta \setminus (C)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(C_1 \sqcap C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & (C_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cap (C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(C_1 \sqcup C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & (C_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cup (C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(\exists R.C)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \{d \in \Delta \mid \exists d'.(d,d') \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
 & & \text{and } d' \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}\} \\
(\forall R.C)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \{d \in \Delta \mid \forall d'.\ \text{if } (d,d') \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
 & & \text{then } d' \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}\} \\
(\mathbf{K}C)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \bigcap_{\mathcal{J} \in \mathcal{M}} (C)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(\mathbf{A}C)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \bigcap_{\mathcal{J} \in \mathcal{N}} (C)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(P)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & P^{\mathcal{V}(\mathcal{I})} \\
(P_1 \sqcap \ldots \sqcap P_n)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & (P_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cap \ldots \cap (P_n)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(\mathbf{K}R)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \bigcap_{\mathcal{J} \in \mathcal{M}} (R)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(\mathbf{A}R)^{\mathcal{I},\mathcal{M},\mathcal{N}} & = & \bigcap_{\mathcal{J} \in \mathcal{N}} (R)^{\mathcal{J},\mathcal{M},\mathcal{N}}
\end{array}
$$

Intuitively, an individual $d \in \Delta$ is an instance of a concept C iff $d \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$ in the particular interpretation $\mathcal{I}$. An individual $d \in \Delta$ is an instance of a concept $\mathbf{K}C$ (i.e. $d \in (\mathbf{K}C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$) iff $d \in C^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for all interpretations $\mathcal{J} \in \mathcal{M}$. That is, d is "known" to be an instance of concept C if it belongs to the concept interpretation of each possible world in $\mathcal{M}$. An individual $d \in \Delta$ is an instance of a concept $\neg\mathbf{A}C$ (i.e. $d \in (\neg\mathbf{A}C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$) iff $d \in \neg C^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for at least one interpretation $\mathcal{J} \in \mathcal{N}$. Namely, an individual is "by default" not an instance of a concept if it belongs to the concept interpretation in at least one possible world of $\mathcal{N}$. Similarly, an individual $d \in \Delta$ is an instance of a concept $\exists\mathbf{K}R.\top$ iff there is an individual $d' \in \Delta$ such that $(d,d') \in R^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for all $\mathcal{J} \in \mathcal{M}$.

An inclusion assertion $C \sqsubseteq D$ is satisfied in $(\mathcal{I},\mathcal{M},\mathcal{N})$ iff $(C)^{\mathcal{I},\mathcal{M},\mathcal{N}} \subseteq (D)^{\mathcal{I},\mathcal{M},\mathcal{N}}$, while an instance assertion $C(a)$ is satisfied in $(\mathcal{I},\mathcal{M},\mathcal{N})$ iff $a \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$ and $R(a,b)$ is satisfied in $(\mathcal{I},\mathcal{M},\mathcal{N})$ iff $(a,b) \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}}$. An inclusion $C \sqsubseteq D$ is *satisfied* by a structure $(\mathcal{M},\mathcal{N})$ iff each interpretation $\mathcal{I} \in \mathcal{M}$ is such that $(\mathcal{I},\mathcal{M},\mathcal{N})$ satisfies $C \sqsubseteq D$. An assertion $C(a)$ (resp. $R(a,b)$) is satisfied by $(\mathcal{M},\mathcal{N})$ iff each interpretation $\mathcal{I} \in \mathcal{M}$ is such that $(\mathcal{I},\mathcal{M},\mathcal{N})$ satisfies $C(a)$ (resp. $R(a,b)$). An $\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base Σ is *satisfied* by a structure $(\mathcal{M},\mathcal{N})$ iff each sentence (inclusion or membership assertion) of Σ is satisfied by $(\mathcal{M},\mathcal{N})$.

Then, a preference semantics is defined over the structures satisfying Σ. Precisely, a set of interpretations $\mathcal{M}$ is a *model* for Σ iff the structure $(\mathcal{M},\mathcal{M})$ satisfies Σ and, for each set of interpretations $\mathcal{M}'$, if $\mathcal{M} \subset \mathcal{M}'$ then $(\mathcal{M}',\mathcal{M})$ does not satisfy Σ.

The $\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base Σ is *consistent* if there exists a model for Σ, *inconsistent* otherwise. Σ logically implies an inclusion assertion $C \sqsubseteq D$, denoted as $\Sigma \models C \sqsubseteq D$, if $C \sqsubseteq D$ is true in each model for Σ. Analogously, *instance checking* in Σ of an assertion $C(a)$ is defined as follows: $\Sigma \models C(a)$ iff $C(a)$ is satisfied by each model of Σ.