A Theory and Implementation of Cognitive Mobile Robots

GIUSEPPE DE GIACOMO, LUCA IOCCHI, DANIELE NARDI and RICCARDO ROSATI, *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy*

Abstract

We describe an approach to reasoning agents which is based on a formal theory of actions and is actually implemented on a mobile robot working in an office environment. From an epistemological viewpoint, our proposal is originated by the correspondence between Dynamic Logics and Description Logics. Specifically, we consider an epistemic extension of Description Logics to provide a new theoretical framework for the representation of dynamic systems, where the agent's reasoning is based on its knowledge about the world. In this setting, we obtain a weaker notion of logical inference, thus simplifying the reasoning task. From a practical viewpoint, we use a general purpose knowledge representation system based on Description Logics and its associated reasoning tools, in order to plan the actions of the mobile robot "Tino", starting from the knowledge about the environment and the action specification. In addition, we exploit the robot's capabilities in order to integrate the execution of the plan with reactive behaviors, thus enabling the agent to accomplish its tasks in the real world.

Keywords: reasoning about actions, cognitive robotics, description logics.

1 Introduction

Research on mobile robots has been developed within the field of Artificial Intelligence by taking the view that a robot is an autonomous agent capable of achieving a variety of goals in a dynamic real environment. Initially, the focus of this research has been on the high-level representation of actions that the robot can perform. However, it soon became clear that it is very difficult to build robots that exhibit the desired behavior in real environments, simply on the basis of such a declarative representation. Consequently, research split into two streams, that developed rather independently of each other. On the one hand, the basic functionalities of the robot, such as navigation or sensor interpretation, have been addressed; on the other hand, sophisticated logic-based representations of the agent have been developed.

Recent work (see for example [8]) has shown that a mobile robot can effectively be provided with reactive capabilities. However, a mobile robot needs not only the ability to promptly react and adjust its behavior based on the information acquired through its sensors, but also to achieve high-level goals. Therefore, it should also be able to reason about the actions it can perform, find plans that allow it to achieve its goals and check whether the execution of actions leads to the accomplishment of the goals. The integration of reactive and planning capabilities has thus become a focus of research in mobile robots and planning systems (see for example [41, 24, 47]).

We believe that this renewed effort to combine a logic-based view of the robot as an intelligent agent with its reactive functionalities is essential to devise agents that operate in a real world environment. In this paper we present a proposal for reconciling a logic-based view of the agent with the implementation of real robots. Specifically, we provide a framework for reasoning about actions and discuss its implementation, through a knowledge-based system, on a robot with reactive capabilities. Thus, our approach falls in the research stream of logic-based approaches for reasoning about actions (see [32]), however it has been developed as a balance between theoretical and practical considerations.

Our proposal for reasoning about actions is originated by the correspondence between Description Logics and Propositional Dynamic Logics (PDLs) [44, 14]. PDLs, that have been developed for reasoning about programs, have also been considered for reasoning about actions (e.g. [40, 15]). In this setting, a dynamic system is represented without explicitly introducing in the language terms to denote states, as, for example in the Situation Calculus. Formulae denote then properties of states, and actions denote state transitions from one state to another. The dynamic system itself is described by two kinds of axioms: "static axioms", that account for background knowledge, and "dynamic axioms", that describe how the state changes when an action is performed. As in the deductive-planning tradition, a plan can be generated by finding a constructive existence proof for a state where the desired goal is satisfied.

Description Logics (DLs) have been developed to support a rational design of knowledge representation languages and their associated reasoning services, centered around the notions of frame and hierarchical organization of knowledge, and constitute the basis of several implemented tools [48]. The key idea underlying our proposal is to exploit the features for knowledge representation and reasoning of DLs to obtain a principled implementation of reasoning about actions in a setting derived from PDLs. In particular, we rely on a DL enriched with an epistemic operator, originally introduced to formally characterize several practical features of knowledge representation systems based on DLs, not captured by a first-order setting [17]. Through the epistemic operator one can distinguish what is true in the world from what is known by the agent, thus allowing both for a new characterization of the dynamic system and for a weaker notion of inference.

Specifically, the work on computational aspects of reasoning in DLs (see [19] for a survey) shows that the typical form of dynamic axioms leads to cyclic assertions in the knowledge base and thus requires sophisticated reasoning techniques. Hence, we have reinterpreted dynamic axioms by means of the so-called procedural rules. By relying on the epistemic interpretation of these rules given in [18], we have defined a setting which provides both an epistemic representation of dynamic axioms and a weak form of reasoning. In this way, we obtain a simplified reasoning task and a semantically justified approach to deductive planning.

We have built an implementation on top of the mobile robot *Erratic* [29], equipped with wheels and sonars, which has the capability of integrating action execution and reactive behavior; we named our mobile robot "Tino" [12]. In particular, we have addressed the implementation of the theory of action and its integration within a real robot architecture. We relied on CLASSIC [7], a well-known, general-purpose knowledge representation system based on DLs. An interesting feature of our approach is that the reasoning tools provided by such a system can be effectively used in the implementation of the theoretical framework. With regard to the integration of the planning component within the robot control software, we relied on the capabilities offered by a built-in fuzzy controller [30] to combine the execution of high-level actions with reactive behaviors that are needed in a real environment. It is worth emphasizing that this kind of integration has been of critical importance for achieving an implementation that has been successfully tested in real office environments.

The paper is organized as follows. In Section 2, we present the basic Description Logic formalism, and its epistemic extension. In Section 3 we describe the general framework for the representation of dynamic systems, and in Section 4 we address our specific proposal for representing and reasoning about actions. In Section 5, we describe the implementation on the mobile robot "Tino", which includes the implementation in CLASSIC. We conclude the paper by discussing related work (Section 6) and drawing some conclusions (Section 7).

2 Epistemic Description Logics

The technical background of our proposal is constituted by Description Logics (DLs) enriched with an epistemic operator. We refer to [19] for a deeper introduction to DLs.

DLs have been developed to represent the domain of interest in terms of *concepts* (unary predicates) that characterize subsets of the objects, called *individuals*, in the domain, and *roles* (binary predicates) over such domain. Concepts are described by means of concept expressions that can be constructed through set operators plus special constructors involving roles that link the individuals belonging to a concept to those of other concepts. One can establish a hierarchy of concepts, based on set containment (called *subsumption*).

We focus on a well-known DL, $\mathcal{ALC}$, [45] and its epistemic extension, $\mathcal{ALCK}$, obtained by adding a modal operator interpreted in terms of minimal knowledge as in [17, 18, 20]. Such an extension plays a critical role in our formalization of dynamic systems.

The abstract syntax of $\mathcal{ALC}$ is defined as follows:

$$C, D \quad ::= \quad A \mid \bot \mid \top \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \forall R.C$$

where A denotes an atomic concept, C and D denote generic concepts, R denotes an atomic role.

We shall give the semantics of $\mathcal{ALC}$ using the so called Common Domain Assumption (CDA): every interpretation is defined over the same, fixed, countable-infinite domain of individuals Δ. CDA is not essential for $\mathcal{ALC}$. It can be easily shown that CDA does not affect the reasoning tasks considered in non-epistemic DLs. However, CDA is required for $\mathcal{ALCK}$.

An $\mathcal{ALC}$*-interpretation* $\mathcal{I}$ as a function mapping each concept expression into a subset of Δ and each role expression into a subset of $\Delta \times \Delta$, such that:

$$\begin{array}{rcl}
A^{\mathcal{I}} & \subseteq & \Delta \\
R^{\mathcal{I}} & \subseteq & \Delta \times \Delta \\
\top^{\mathcal{I}} & = & \Delta \\
\bot^{\mathcal{I}} & = & \emptyset \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta \setminus C^{\mathcal{I}} \\
\forall R.C^{\mathcal{I}} & = & \{d_1 \in \Delta \mid \forall d_2.(d_1, d_2) \in R^{\mathcal{I}} \Rightarrow d_2 \in C^{\mathcal{I}}\} \\
\exists R.C^{\mathcal{I}} & = & \{d_1 \in \Delta \mid \exists d_2.(d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}
\end{array}$$

DLs are typically used for representing the knowledge about a problem domain by providing mechanisms for specifying relationships among concepts, and for providing information about specific individuals. Accordingly, an $\mathcal{ALC}$ *knowledge base* Σ is defined as a pair

$\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$, called the *TBox*, is a finite set of inclusion assertions of the form $C \sqsubseteq D$, with $C, D \in \mathcal{ALC}$, and $\mathcal{A}$, called the *ABox*, is a finite set of membership assertions of the form $C(a)$ or $R(a, b)$, where $C, R \in \mathcal{ALC}$ and a, b are *names* of individuals. We assume that different names denote different individuals, and with a slight abuse of notation, we do not distinguish between individuals and their names.

The semantics of inclusion assertions is defined in terms of set inclusion: $C \sqsubseteq D$ is satisfied in $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Membership assertions are interpreted in terms of set membership: $C(a)$ is satisfied in $\mathcal{I}$ iff $a \in C^{\mathcal{I}}$ and $R(a, b)$ is satisfied in $\mathcal{I}$ iff $(a, b) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies an $\mathcal{ALC}$-knowledge base Σ iff every inclusion and membership assertion of Σ is satisfied in $\mathcal{I}$. A knowledge base Σ *logically implies* an inclusion or membership assertion σ, written $\Sigma \models \sigma$, iff every interpretation $\mathcal{I}$ satisfying Σ also satisfies σ.

The description logic $\mathcal{ALCK}$ is an extension of $\mathcal{ALC}$ with an epistemic operator interpreted as (minimal) knowledge. More precisely the $\mathcal{ALCK}$ abstract syntax is as follows:

$$\begin{array}{lcl} C, D & ::= & A \mid \top \mid \bot \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \forall Q.C \mid \exists Q.C \mid \mathbf{K}C \\ Q & ::= & R \mid \mathbf{K}R \end{array}$$

where A denotes an atomic concept, C and D denote generic concepts, R denotes an atomic role, and Q a generic role.

Non-epistemic concepts and roles are given the standard semantics of DLs, while epistemic concepts and roles are interpreted on the class of Kripke structures where worlds are $\mathcal{ALC}$-interpretations, and all worlds are connected to each other, i.e. the accessibility relation among $\mathcal{ALC}$-interpretations is universal, thus Kripke structures correspond to sets of $\mathcal{ALC}$ interpretations.

An *$\mathcal{ALCK}$-interpretation* is defined as a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{W}$ is a set of $\mathcal{ALC}$-interpretations over the domain Δ, and $\mathcal{I}$ is a distinguished interpretation belonging to $\mathcal{W}$ (i.e. $\mathcal{I} \in \mathcal{W}$), such that:

$$\begin{array}{rcl} A^{\mathcal{I},\mathcal{W}} & \subseteq & \Delta \\ R^{\mathcal{I},\mathcal{W}} & \subseteq & \Delta \times \Delta \\ \top^{\mathcal{I},\mathcal{W}} & = & \Delta \\ \bot^{\mathcal{I},\mathcal{W}} & = & \emptyset \\ (C \sqcap D)^{\mathcal{I},\mathcal{W}} & = & C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \\ (C \sqcup D)^{\mathcal{I},\mathcal{W}} & = & C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}} \\ (\neg C)^{\mathcal{I},\mathcal{W}} & = & \Delta \setminus C^{\mathcal{I},\mathcal{W}} \\ (\forall Q.C)^{\mathcal{I},\mathcal{W}} & = & \{d_1 \in \Delta \mid \forall d_2. (d_1, d_2) \in Q^{\mathcal{I},\mathcal{W}} \Rightarrow d_2 \in C^{\mathcal{I},\mathcal{W}}\} \\ (\exists Q.C)^{\mathcal{I},\mathcal{W}} & = & \{d_1 \in \Delta \mid \exists d_2. (d_1, d_2) \in Q^{\mathcal{I},\mathcal{W}} \wedge d_2 \in C^{\mathcal{I},\mathcal{W}}\} \\ (\mathbf{K}C)^{\mathcal{I},\mathcal{W}} & = & \bigcap_{\mathcal{J} \in \mathcal{W}} (C^{\mathcal{J},\mathcal{W}}) \\ (\mathbf{K}R)^{\mathcal{I},\mathcal{W}} & = & \bigcap_{\mathcal{J} \in \mathcal{W}} (R^{\mathcal{J},\mathcal{W}}). \end{array}$$

Intuitively, an individual $d \in \Delta$ is an instance of a concept C iff $d \in C^{\mathcal{I},\mathcal{W}}$ in the particular $\mathcal{ALC}$-interpretation $\mathcal{I} \in \mathcal{W}$. An individual $d \in \Delta$ is an instance of a concept

$\mathbf{K}C$ (i.e. $d \in (\mathbf{K}C)^{\mathcal{I},\mathcal{W}}$) iff $d \in C^{\mathcal{J},\mathcal{W}}$ for all possible $\mathcal{ALC}$-interpretations $\mathcal{J} \in \mathcal{W}$. The individual $d \in (\mathbf{K}C)^{\mathcal{I},\mathcal{W}}$ is said to be a *known instance* of a concept C; also, C is said to be *known* for the individual d. 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 $(d, d') \in R^{\mathcal{J},\mathcal{W}}$ for all possible $\mathcal{J} \in \mathcal{W}$. The individual d' is said to be a *known* R-successor of d.

An $\mathcal{ALCK}$ knowledge base Σ is defined as a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, where the TBox $\mathcal{T}$ is a finite set of inclusion assertions of the form $C \sqsubseteq D$, with $C, D \in \mathcal{ALCK}$, and the ABox $\mathcal{A}$ is a finite set of membership assertions of the form $C(a)$ or $R(a, b)$, with $C, R \in \mathcal{ALCK}$ and a, b names of individuals. $C \sqsubseteq D$ is satisfied in $(\mathcal{I}, \mathcal{W})$ iff $C^{\mathcal{I},\mathcal{W}} \subseteq D^{\mathcal{I},\mathcal{W}}$. $C(a)$ is satisfied in $(\mathcal{I}, \mathcal{W})$, iff $a \in C^{\mathcal{I},\mathcal{W}}$ and $R(a, b)$ is satisfied in $(\mathcal{I}, \mathcal{W})$ iff $(a, b) \in R^{\mathcal{I},\mathcal{W}}$. A *model* for an $\mathcal{ALCK}$-knowledge base Σ is a set of $\mathcal{ALC}$-interpretations $\mathcal{W}$ such that for each interpretation $\mathcal{I} \in \mathcal{W}$, every inclusion and membership assertion of Σ is satisfied in the $\mathcal{ALCK}$-interpretation $(\mathcal{I}, \mathcal{W})$.

Next, a preference semantics on universal Kripke structures is defined, which allows one to select only those models where the knowledge is minimal. This is achieved by maximizing in each epistemic model the number of possible worlds (i.e. $\mathcal{ALC}$-interpretations), which can also be explained as maximizing ignorance.

A *preferred model* $\mathcal{W}$ for Σ is a model for Σ such that $\mathcal{W}$ is a maximal set of $\mathcal{ALC}$-interpretations, in the sense that for each set $\mathcal{W}'$, if $\mathcal{W} \subset \mathcal{W}'$ then $\mathcal{W}'$ is not a model for Σ. Σ is *satisfiable* if there exists a preferred model for Σ, *unsatisfiable* otherwise. Σ *logically implies* an (inclusion or membership) assertion σ, written $\Sigma \models \sigma$, iff σ is satisfied in every preferred model for Σ.

By using the epistemic operator, it is possible to formalize in $\mathcal{ALCK}$ several practical features provided by implemented knowledge representation systems based on DLs [18]. In particular, here we recall the so-called *procedural rules* (or simply rules).

Procedural rules take the form:

$$C \mapsto D$$

(where C, D are $\mathcal{ALC}$ concepts). Roughly speaking, their meaning is "if an individual is proved to be an instance of C, then conclude that it is also an instance of D". They can be viewed as a weak form of implication for which the contrapositive does not hold. A procedural rule $C \mapsto D$ can be formalized in $\mathcal{ALCK}$ by the epistemic assertion

$$\mathbf{K}C \sqsubseteq D$$

A consistent knowledge base in which the epistemic operator occurs only in rules of the above form has a unique preferred model; moreover, in such a case reasoning can be accomplished by constructing a knowledge base, called *first-order extension* [17, 18].

Finally, we address computational aspects of reasoning in DLs. Research in DLs has shown that there is a trade-off between the set of concept forming constructs allowed in a DL and the complexity of reasoning. More specifically, two aspects must be carefully considered: the set of constructs allowed in the expressions; The form of the inclusion assertions.

In particular, it has been shown [45] that logical implication in $\mathcal{ALC}$ is an EXPTIME-complete problem. The high complexity of logical implication in $\mathcal{ALC}$ is due to the general form of inclusion assertions allowed. When considering general inclusion assertions of the form $C \sqsubseteq D$, reasoning becomes EXPTIME-hard even for a simple language as $\mathcal{FL}_0$, [36], subset of $\mathcal{ALC}$ which contains only intersection $\sqcap$ and universal quantification $\forall$. Hence, restrictions on the form of inclusion assertions are normally considered. In particular, a set of inclusion assertions is cyclic, when a chain of individuals connected through roles can be built that connects a concept with itself. Cycles are especially problematic from the computational point of view [37, 10, 11], and typically are not allowed by implemented systems.

Logical implication of the form $\Sigma \models C(a)$ is called *instance checking*. Logical Implication, and in particular instance checking in the case of $\mathcal{ALC}$ knowledge bases with acyclic inclusion assertions is PSPACE-complete [19], and we get the same complexity characterization even if we consider an empty knowledge base. Moreover, instance checking remains PSPACE-complete, even when $\mathcal{ALCK}$ is used to express the query concept C and procedural rules are allowed in the knowledge base [17].

3 Formalization of Dynamic Systems

Dynamic systems are typically modeled in terms of state transitions caused by actions. A *state* represents a possible state of affairs of the system. Each state is associated with a set of properties which hold in the represented state of affairs. *Actions* cause state transitions, making the system evolve from the current state to the next one.

In principle the *behavior* of a system, i.e. all its possible evolutions, can be represented as a *transition graph*, where each node denotes a state and is labeled with its associated properties, and each arc denotes a state transition and is labeled with the action that causes the transition. Note, however, that *complete knowledge* of the behavior of the system is required to actually define its transition graph, while in general one has only partial knowledge of such behavior. In deductive planning this knowledge is phrased in terms of *axioms* of some logic (e.g. the Situation Calculus [39] or PDLs [40]). These axioms select a subset of all possible transition graphs. The actual behavior of the system is in fact captured by one of the selected graphs. However, since typically one does not have enough knowledge to isolate the specific graph capturing the actual behavior, one has to focus on those properties that are true in all the selected graphs, i.e. those properties that are *logically implied* by the axioms.

Our general framework allows for specifying the behavior of the dynamic system by means of three kinds of axioms: "static axioms", action "precondition axioms", and "effect axioms".

- *Static axioms* specify properties which are true in every state and do not depend on actions. In other words, static axioms are used for representing background knowledge that is invariant with respect to the execution of actions.
- *Precondition axioms* specify circumstances under which it is possible to execute an action. We assume that the designer of the system is able to specify *sufficient conditions* for actions to be executed.
- *Effect axioms* specify (direct) *effects* of an action if executed under given circumstances, i.e. if executed in a state satisfying certain *premises*. Obviously, through static axioms additional effects can be inferred from those specified by effect axioms. Observe that in general we do not require the designer to specify all the effects of an action.

The last two kinds of axioms are referred to as *dynamic axioms*, since both are used to model dynamic aspects of the system.

In addition to these kinds of axioms, we assume that the designer specifies the knowledge on the initial state, by providing an *initial state description* in terms of the properties (not involving actions) that are associated with the initial state.

Actions are assumed to be *deterministic*, which means that in each transition graph, given a state and an action, at most a single successor state is determined. However, the properties associated with such successor state are generally different in different transition graphs. We might say that actions are deterministic but their effects are *underdetermined* in general.

To illustrate the framework, we first show how the various parts of the specification of a dynamic system can be expressed in the Situation Calculus [39]. The Situation Calculus is a dialect of classical logic, in which two term sorts are distinguished: *situation*-terms that are used to denote states[1], and *action*-terms that denote actions. Predicates that denote properties associated with states have a special situation argument and are called *fluents*.

Following the general framework above, the behavior of the dynamic system can be specified as follows:

- *Static axioms* can be expressed as

  $$\forall s.\phi(s)$$

  where $\phi(s)$ is a formula in which s is the unique situation argument of all fluents. The axiom specifies that ϕ holds in every state.

- *Precondition axioms* can be expressed as

  $$\forall s.\phi(s) \Rightarrow Poss(r, s)$$

  where $Poss(r, s)$ is a special predicate that denotes whether it is possible to execute the action r in s, and $\phi(s)$ is a formula in which s is the unique situation argument of all fluents and in which $Poss$ does not occur. The axiom specifies that if the precondition ϕ holds in s then r can be executed in s.

- *Effect axioms* can be expressed as

  $$\forall s.Poss(r, s) \wedge \phi(s) \Rightarrow \psi(do(r, s))$$

  where $do(r, s)$ is a situation that denotes the state resulting from executing r in s, and $\phi(s)$, $\psi(do(a, s))$ are formulae in which s and $do(r, s)$, respectively, are the unique situation argument of all fluents and in which $Poss$ does not occur. The axiom specifies that in the state resulting from executing r in s, provided that r is executable, ψ holds if the premise ϕ is satisfied in s.

The *initial state description* can be expressed as a sentence of the form

$$\phi(s_0)$$

where s_0 is a special situation that denotes the initial state, and $\phi(s_0)$ is a formula in which s_0 is the unique situation argument of all fluents. The sentence specifies that ϕ holds in the initial state.

[1]To avoid confusion, observe that what we call here states are essentially the semantic counterparts of situation-terms. Instead sometimes in the Situation Calculus the word state is used to refer to an interpretation of all atomic fluents in a situation.

In DLs *states* are represented by *individuals*, i.e. elements of the domain of interpretation. *Properties of states* are represented by *concepts*. *Actions*, which are assumed to be deterministic, are represented as *functional roles*, i.e. roles interpreted as functions instead of relations. In fact, we distinguish two kinds of roles: *static-roles*, which represent the usual notion of role in DLs and can be used for structuring properties of states, and *action-roles*, which are functional roles that denote actions. When R is an action-role, the construct $\exists R.\top$ expresses that there exists an execution of the action R, i.e. R is possible in the current state. Instead, the construct $\forall R.C$ expresses that all executions of R lead to a state where C holds, i.e., since R is deterministic, if the action R is executable then it leads to a state where C holds.

The behavior of the dynamic system is specified in terms of ($\mathcal{ALC}$) inclusion assertions as follows:

- *Static axioms*
$$C \sqsubseteq D$$
where C and D are concepts not involving action-roles. The assertion expresses that every state satisfying the property denoted by C satisfies also the property denoted by D.

- *Precondition axioms*
$$C \sqsubseteq \exists R.\top$$
where C is a concept representing a property of the current state that is sufficient in order to execute the action R. C does not involve action-roles. Multiple axioms per action are allowed.

- *Effect axioms*
$$C \sqsubseteq \forall R.D$$
where D is a concept representing a property that holds in the state resulting from executing R in a state satisfying the property C. C and D do not involve action-roles. Multiple axioms per action are allowed.

We denote with Γ the set of axioms in the specification. The specification apart from Γ includes the *initial state description*, which is expressed by a concept S (not a set of assertions).

Observe that a major difference between the formalization of dynamic systems in the Situation Calculus and the one in DLs, is that in the former there is an explicit denotation of states through situation-terms, while in the latter states are implicitly described through their properties. In this respect, the DL-based formalization is analogous to a modal formalization of dynamic systems.

Indeed, the formalization in DLs is strictly related to those based on Propositional Dynamic Logics (PDLs) [40, 15]. PDLs [23] are modal logics for reasoning about computer programs that have been studied extensively (see [31] for a survey). It has been shown that there is a tight correspondence between DLs and PDLs [44, 14], which allows for considering PDLs and DLs as syntactic variants of each other. Fig. 1 shows how the constructs of PDLs can be translated into constructs of DLs and vice versa. In view of this correspondence, the DL-based formalization considered above can be seen as a variant of the one proposed in [40]. The only difference is that in [40] precondition axioms were not considered, forcing each action R to be executable in every state.

PDLs		DLs	
atomic proposition	A	atomic concept	A
true	`tt`	top	$\top$
false	`ff`	bottom	$\bot$
conjunction	$C \wedge D$	conjunction	$C \sqcap D$
disjunction	$C \vee D$	disjunction	$C \sqcup D$
negation	$\neg C$	negation	$\neg C$
diamond ("some executions...")	$\langle r \rangle C$	existential quantification	$\exists R.C$
box ("all executions...")	$[r]C$	universal quantification	$\forall R.C$
valid implication (axiom)	$C \Rightarrow D$	inclusion assertion	$C \sqsubseteq D$
		instance assertion	$C(a) \mid R(a_1, a_2)$

Figure 1: Correspondence between DLs and PDLs.

The kind of reasoning we are interested in can be phrased in the DL-based formalization as a logical implication of the form

$$\Gamma \models S \sqsubseteq D \tag{1}$$

which expresses that, given a specification of the behavior of the system Γ, and an initial state described by S, it follows that the initial state also satisfies the property denoted by D, possibly involving the execution of actions.

Specifically, in deductive planning one is interested in answering the following question: "Is there a sequence of actions that, starting from an initial state, leads to a state where a given property (the goal) holds?". Under the assumption of deterministic actions, this is captured by the following logical implication:

$$\Gamma \models S \sqsubseteq PLAN\_FOR\_G \tag{2}$$

where $PLAN\_FOR\_G$ stands for any concept of the form $\exists R_1.\exists R_2.\ldots\exists R_n.G$ (with $n \geq 0$ and R_i any action), and expresses the existence of a finite sequence of actions leading to a state where the goal G is satisfied. Formally, $PLAN\_FOR\_G$ can be defined inductively as the concept $\mathcal{P}_{\mathcal{S}}$ such that: $G \sqsubseteq \mathcal{P}_{\mathcal{S}}$; if $C \sqsubseteq \mathcal{P}_{\mathcal{S}}$, then $\exists R.C \sqsubseteq \mathcal{P}_{\mathcal{S}}$, for every action role R.[2] From a *constructive proof* of the above logical implication one can extract an actual sequence of actions (a plan) that leads to the goal.

Observe that in this setting one may have very sparse knowledge about the system – say a few laws (axioms) one knows the system obeys – and yet be able to make several non-trivial inferences. Unfortunately, this generality leads to a high computational cost, as shown by the following proposition, where by propositional concept we refer to those concept expressions that are formed excluding quantifications on roles.

Proposition 1 Consider a specification Γ of a dynamic system having the following characteristics:

- Static axioms of the form $C \sqsubseteq D$, with C and D propositional.
- Precondition axioms of the form $C \sqsubseteq \exists R.\top$, with C propositional.

[2] We use $\exists R.C$ as an abbreviation for $\exists R_i.\top \sqcap \forall R_i.C$. Indeed, since actions are assumed to be deterministic, the two concept expressions are equivalent.

- Effect axioms of the form $C \sqsubseteq \forall R.D$, with C,D propositional.
- Initial state description constituted by a concept S which is propositional.

Deciding logical implications $\Gamma \models S \sqsubseteq D$, with D propositional, is an EXPTIME-hard problem.

The proposition is an easy consequence of a complexity result in [11]: reasoning with primitive inclusion assertions, i.e. inclusion assertions of the form $A \sqsubseteq C$ with A an atomic concept, in the DL $\mathcal{ALU}$, obtained from $\mathcal{ALC}$ by restricting existential qualification to the form $\exists R.\top$, is EXPTIME-hard. In fact, the same complexity bound holds if we restrict the class of specifications to those having precondition axioms of the form $\top \sqsubseteq \exists R.\top$, as in [40].

Note that in the proposition above static axioms are trivially acyclic, since they are propositional. Instead, the cyclicity of dynamic axioms is unavoidable, otherwise we could not relate the truth-value of a property in state resulting from executing an action to that in the current state. For example we could not express that a property C persists over an action R, i.e. $C \sqsubseteq \forall R.C$.

Finally, observe that the framework presented here is quite standard both in Artificial Intelligence and in Computer Science. However, it does not deal directly with well known formalization problems studied in Artificial Intelligence (see e.g. [42]). In particular, the *qualification problem* – i.e. the problem of finding out the actual conditions that must be true to execute an action – does not arise since the designer is required to specify sufficient conditions for the execution of an action. While the *frame problem* – i.e. the problem of specifying what remains unchanged when an action is executed – is dealt with by requiring the designer to specify explicitly what remains unchanged by using frame axioms, that is effect axioms of the form $C \sqsubseteq \forall R.C$. Note that the designer is allowed to specify effects of actions partially. Furthermore, static axioms combined with effect axioms can be used to specify *ramifications*, i.e. indirect effects.

4 Reasoning about actions in epistemic DLs

In the framework described in the previous section, the dynamics of the system is specified in terms of *what is true in the world*. Now, we modify it, obtaining a new framework in which the dynamics is specified in terms of *what the robot knows of the world*. The idea is that the robot achieves its conclusions based on its *epistemic state* and not on the actual state of the world.

As we shall see, this change of viewpoint in the representation has two important consequences:

- It simplifies reasoning, and in particular it simplifies the task of deducing plans.
- It makes deductive planning always constructive, i.e. the reachability of a state in which the goal holds is deduced only if there exists a known sequence of actions that leads to it.

Below we introduce the new representation framework and, subsequently, address reasoning.

4.1 Representation

The behavior of the agent is again described by means of both static axioms and dynamic axioms, divided into precondition axioms and effect axioms. Static axioms are inclusion assertions not involving action-roles, as before, and not involving cycles, in order to avoid the difficulties of reasoning about them. Dynamic axioms have a different form that makes use of the epistemic operator $\mathbf{K}$ (in the following C, D denote $\mathcal{ALC}$ concepts):

- *Precondition axioms* have the form

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \tag{3}$$

  which is interpreted as: if a state (individual) x is an instance of C in all possible interpretations, then there exists a state y, the same in all possible interpretations, which is the (unique) $\mathbf{K}R$-successor of x.

- *Effect axioms* have the following form

$$\mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{K}D \tag{4}$$

  that can be read as: if a state (individual) x is an instance of C in all possible interpretations, then in every interpretation each $\mathbf{K}R$-successor of x (in fact at most one, being actions deterministic) is a known instance of D.

A special form of effect axioms considered above corresponds to *frame axioms*, i.e. axioms expressing which properties do not change when certain actions are executed. Indeed by writing

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

we express the fact that the property C is not affected by the execution of action R.

Also the *initial state description* is given in a different way. We denote by the individual name $init$ the initial state, and give the initial state description in terms of a membership assertion on $init$. In other words, the initial state description has the form

$$S(init)$$

where S is a concept that expresses the known properties of the initial state, and $init$ is an individual name that denotes the initial state.

The above formalization differs from the one given in Section 3 in two major aspects. First, we have introduced a mechanism to denote states independently of the interpretation of the concepts, namely chains of any length of roles of the form $\mathbf{K}R$, with R an action-role, starting from $init$. In fact, $init$ denotes an individual which in every interpretation represents the initial state. Similarly, the $\mathbf{K}R_1$-successor of $init$ is an individual $x_{init;R_1}$ which represents in every interpretation the $\mathbf{K}R_1$-successor of the state represented by $init$. The $\mathbf{K}R_2$-successor of $x_{init;R_1}$ is an individual $x_{init;R_1;R_2}$ which represents in every interpretation the $\mathbf{K}R_2$-successor of the state represented by $x_{init;R_1}$. And so on for chains of any length of roles of the form $\mathbf{K}R$, starting from $init$. As observed, the above described mechanism is not available in non-epistemic DLs as well as in PDLs, although it is provided by the Situation Calculus through situation-terms.

Secondly, the possibility of executing an action, and the effects that are obtained by executing that action, are given in terms of *what is known* about the current state. In fact, the presence of $\mathbf{K}C$, instead of C, in the left-hand side of both precondition axioms and effect axioms, requires C to be known for the individual x representing the current state, or more precisely requires C to be true for the state denoted by the individual x in all interpretations, i.e. C is required to be *valid* for the individual x. As mentioned in Section 2, this use of $\mathbf{K}$ is tightly connected with the realization of procedural rules in epistemic DLs. Observe that, for every property C, either C is valid for an individual or not, i.e. either C is known or it is not known in the current state. Namely, the robot has *complete* knowledge on its epistemic state. Such epistemic state in turn expresses the *partial* knowledge the robot has on the actual current state.

Let us now look at the planning problem. Let Γ be the set of static axioms, precondition axioms, and effect axioms, specifying the behaviour of the dynamic system. Given an initial state $init$ satisfying certain properties S, a plan exists for a specified goal iff there exists a finite sequence of actions that, from the initial state $init$, leads to a state satisfying the goal. This condition is expressed by a logical implication similar to (2), namely:

$$\langle \Gamma, \{S(init)\}\rangle \models PLAN\_FOR\_G(init) \tag{5}$$

where $PLAN\_FOR\_G$ is the concept $\mathcal{P}_{\mathcal{S}}$ defined inductively as: $\mathbf{K}G \sqsubseteq \mathcal{P}_{\mathcal{S}}$; if $C \sqsubseteq \mathcal{P}_{\mathcal{S}}$, then $\exists \mathbf{K}R_i.C \sqsubseteq \mathcal{P}_{\mathcal{S}}$, for every action-role R_i. In other words, $PLAN\_FOR\_G$ stands for any concept expression of the form $\exists \mathbf{K}R_1.\exists \mathbf{K}R_2.\ldots.\exists \mathbf{K}R_n.\mathbf{K}G$ in which $n \geq 0$ and each R_i is an action-role, and it expresses the fact that from the initial state $init$ there exists a sequence of successors (the same in every interpretation) that terminates in a state (the same in every interpretation) where G holds (in every interpretation).

Condition (5) holds iff for each preferred model $\mathcal{W}$ for $\Sigma = \langle \Gamma, \{S(init)\}\rangle$, there exists a state $x \in \Delta$ such that $x \in G^{\mathcal{I},\mathcal{W}}$ for all $\mathcal{I} \in \mathcal{W}$. Indeed, due to the special form of the dynamic axioms, such a state exists iff it is linked to the initial state by a chain of $\mathbf{K}R_i$, i.e. if there exists a sequence of successors (the same in every possible interpretation) that terminates in x.

We next show, through a simple example, that the use of epistemic inclusion assertions in the formalization of the dynamic system actually weakens the deductive capabilities of the robot. Let us first focus on the non-epistemic framework. Let Γ be the following specification of a dynamic system behavior:

Precond. axioms:	*Effect axioms:*
$C_1 \sqcap C_2 \sqsubseteq \exists R_1.\top$	$C_1 \sqsubseteq \forall R_1.D$
$C_1 \sqcap \neg C_2 \sqsubseteq \exists R_2.\top$	$C_1 \sqsubseteq \forall R_2.D$

and the initial state description be C_1. Suppose we want to know whether exists a plan for achieving D, i.e.:

$$\Gamma \models C_1 \sqsubseteq PLAN\_FOR\_D$$

It is easy to see that the answer to this planning problem is yes, since $\Gamma \models C_1 \sqsubseteq \exists R_1.D \sqcup \exists R_2.D$.

Now let us consider the epistemic framework. The specification of the dynamic system behavior becomes Γ':

Precond. axioms:	*Effect axioms:*
$\mathbf{K}(C_1 \sqcap C_2) \sqsubseteq \exists \mathbf{K}R_1.\top$	$\mathbf{K}C_1 \sqsubseteq \forall \mathbf{K}R_1.\mathbf{K}D$
$\mathbf{K}(C_1 \sqcap \neg C_2) \sqsubseteq \exists \mathbf{K}R_2.\top$	$\mathbf{K}C_1 \sqsubseteq \forall \mathbf{K}R_2.\mathbf{K}D$

and the initial state description: $C_1(init)$. The planning problem becomes:

$$\langle \Gamma', \{C_1(init)\}\rangle \models PLAN\_FOR\_D(init)$$

The answer to this planning problem is no, since neither the robot knows that R_1 is executable, nor it knows that R_2 is executable, hence it has no plan to reach D. The difference is that in the actual state of the world either C_2 is true or $\neg C_2$ is true. However neither C_2 nor $\neg C_2$ is known to be true by the robot (neither of the two is valid), hence the robot concludes that it cannot perform any action.

In fact, there is no *known* sequence of actions leading to the state satisfying the goal, yet in the non-epistemic framework the answer to the planning problem is yes, whereas it is no in the epistemic framework.

The example highlights the differences between the epistemic setting and the non-epistemic one. On the one hand, there are cases in which in the non-epistemic setting the answer to a planning problem is yes, i.e. the logical implication (2) holds, while in the epistemic setting the answer is no, i.e. the logical implication (5) does not hold. Conversely, it is easy to see that, if in the epistemic setting the logical implication (5) holds, then the logical implication (2) holds as well in the non-epistemic setting. That is, reasoning in the epistemic framework is a sound and well characterized approximation of the non-epistemic case. On the other hand, in the epistemic setting only constructive proofs are taken into consideration, in the sense that the logical implication (5) holds only if there exists a *known* sequence of actions leading to the state satisfying the goal. That is, if (5) holds then we are always able to extract an actual plan. As we have seen above this is not always the case in the non-epistemic setting. Hence we can conclude that the use of the epistemic operator in the formalization of actions allows for a *principled weakening* of the deductive capabilities of the robot.

4.2 Reasoning

Let us now turn our attention to the problem of computing the logical implication (5). It is worth noticing that in general the $\mathcal{ALCK}$-knowledge base $\Sigma = \langle \Gamma, \{S(init)\}\rangle$ has many preferred models, which are distinguishable even up to the renaming of states. Nevertheless, due to the special form of the epistemic inclusion assertions corresponding to the dynamic axioms in Σ, we can build the so-called *first-order extension* (FOE) of Σ [18], which consists of the knowledge base $\langle \Gamma_S, \{S(init)\}\rangle$ augmented by a set of membership assertions which are consequences (up to the renaming of states) of the epistemic inclusion assertions.

The FOE of Σ provides a unique characterization of the knowledge that is shared by all the preferred models of Σ. In fact, by representing dynamic axioms by means of epistemic inclusion assertions of special form above, we recover the ability of representing the behavior of the system by means of a *single* graph, corresponding to the FOE. Such a graph is built in the following way:

- Nodes correspond to individuals denoted by chains of $\mathbf{K}R$ of any length starting from $init$ and are labeled by the properties that are known for the corresponding individual.
- There is an edge, labeled by R, from a node x to a node y, if y is an $\mathbf{K}R$-successor of x.

The FOE summarizes the common part of all transition graphs that, by our (partial) knowledge about the dynamic system, are considered possible, thus providing a description of the

ALGORITHM FOE
INPUT: $\Sigma = \langle \Gamma_S \cup \Gamma_P \cup \Gamma_E, \{(init)\}\rangle$
OUTPUT: $FOE(\Sigma)$
begin
STATES = $\{init\}$;
ALL-STATES = $\{init\}$;
$ABOX = \{S(init)\}$;
repeat
s = choose(STATES);
for each action-role R **do**
if $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \in \Gamma_P \wedge \langle \Gamma_S, ABOX\rangle \models C(s)$ **then**
begin
s' = NEW state name;
$ABOX' = ABOX \cup \{R(s,s')\} \cup \{D_i(s') | D_i \in POST(s, R, \langle \Gamma_S, ABOX\rangle, \Gamma_E)\}$;
if there exists a state $s'' \in$ ALL-STATES **such that**
$CONCEPTS(\langle \Gamma_S, ABOX\rangle, s'') = CONCEPTS(\langle \Gamma_S, ABOX'\rangle, s')$
then $ABOX = ABOX \cup R(s, s'')$
else begin
$ABOX = ABOX'$;
STATES = STATES $\cup \{s'\}$
ALL-STATES = ALL-STATES $\cup \{s'\}$
end
end;
STATES = STATES $-\{s\}$
until STATES = $\emptyset$;
return $\langle \Gamma_S, ABOX\rangle$
end;

Figure 2: Algorithm computing $FOE(\Sigma)$

actual transition graph which is partial, in the sense that: (i) certain states and transitions may be missing; (ii) the properties of the states in the graph may be only partially specified.

Again, let $\Gamma = \Gamma_S \cup \Gamma_P \cup \Gamma_E$ be the set of static axioms, precondition axioms, and effect axioms, specifying the behavior of the dynamic system, let $S(init)$ be the specification of the initial state $init$, and let $\Sigma = \langle \Gamma, \{S(init)\}\rangle$. The FOE of Σ, written $FOE(\Sigma)$, is computed by the algorithm shown in Fig. 2, in which

$$POST(s, R, \langle \Gamma_S, ABOX\rangle, \Gamma_E) = \{D \mid \mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{K}D \in \Gamma_E \wedge \langle \Gamma_S, ABOX\rangle \models C(s)\}$$

denotes the set of properties representing the effects (according to the specification Γ_E) of the execution of the action R in the state s, and

$$CONCEPTS(\langle \Gamma_S, ABOX\rangle, s) = \{D \mid \langle \Gamma_S, ABOX\rangle \models D(s)\}$$

denotes the set of properties verified by the state s in Σ.

Informally, the algorithm, starting from the initial state $init$, iteratively proceeds as follows. First, it finds an action R which can be executed in the current state, by identifying in the set Γ_P a precondition axiom for R whose left-hand side is logically implied by the current knowledge base. Then, it propagates the effects of the action R, which again is based on checking whether the left-hand side of each effect axiom for R in the set Γ_E is logically implied by the properties holding in the current state. In this way, the set of properties corresponding to the effect of the execution of R in the current state is computed. A new state is then generated, unless a state with the same properties has already been created. This step is repeated until all actions executable in the current state have been considered. Then, a new current state is chosen among those previously created and the main iteration proceeds, ultimately producing a sort of non-epistemic "completion" of the knowledge base Σ.

The FOE is unique, that is, every order of extraction of the states from the set STATES produces the same set of assertions, up to the renaming of states. Moreover, the algorithm terminates, that is, the condition STATES = $\emptyset$ is eventually reached, since the number of states generated is bounded to the number of different conjunctions of concept expressions in the set $\mathcal{E} = \{D | \mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{K}D \in \Gamma_E\}$, i.e. 2^n, where n is the number of axioms in Γ_E. Finally, the condition

$$CONCEPTS(\langle \Gamma_S, ABOX \rangle, l) = CONCEPTS(\langle \Gamma_S, ABOX' \rangle, j)$$

can be checked by verifying whether for each concept C, obtained as a conjunction of concept expressions in $\mathcal{E}$, $\langle \Gamma_S, ABOX \rangle \models C(l)$ iff $\langle \Gamma_S, ABOX' \rangle \models C(j)$.

We point out that, while the notion of minimization of the properties of states, realized through the propagation of effects (i.e. only the properties which are necessarily implied are propagated), is also correctly captured by the minimal knowledge semantics of $\mathcal{ALCK}$, the minimization obtained in the FOE by always generating one new successor state does not have a direct counterpart in the semantics: This is the reason why Σ has multiple preferred models, whereas the FOE is unique.

The following property establishes that, with respect to the logical implication problem (5), the first-order extension of Σ represents the information which must hold in *any* preferred model for Σ. This property allows one to solve the planning problem (5) by verifying whether there is an x in $FOE(\Sigma)$ that satisfies the goal G.

Theorem 2 There exists a state x such that

$$FOE(\Sigma) \models G(x) \tag{6}$$

if and only if, for each preferred model $\mathcal{W}$ for Σ, there exists a state x such that $x \in G^{\mathcal{I},\mathcal{W}}$ for all $\mathcal{I} \in \mathcal{W}$.

Proof *If-part.* Suppose that for each preferred model $\mathcal{W}$ for Σ there exists a state x such that $x \in G^{\mathcal{I},\mathcal{W}}$ for all $\mathcal{I} \in \mathcal{W}$. Now, due to the preference semantics of $\mathcal{ALCK}$, it follows that there exists a preferred model for Σ which is isomorphic to the unique preferred model of $FOE(\Sigma)$. Hence, there exists a preferred model $\mathcal{W}'$ for Σ such that for each state s in $FOE(\Sigma)$ there exists a state s' in $\mathcal{W}$ isomorphic to s, that is, for every concept C, $FOE(\Sigma) \models C(s)$ iff $s' \in C^{\mathcal{I},\mathcal{W}}$. Therefore, the existence of such a model $\mathcal{W}'$ implies the existence of a state y such that $FOE(\Sigma) \models G(y)$.

Only-if-part. Assume there exists a state x such that $FOE(\Sigma) \models G(x)$. Then, there is a finite sequence of actions $R_{i_1}, \ldots, R_{i_n}$ (the plan) that generates x. Let x_1 be the R_{i_1}-successor of $init$ in $FOE(\Sigma)$, and let $\mathcal{W}$ be any preferred model for Σ. Now, the properties that $init$ is known to verify in all worlds of $\mathcal{W}$ are at least the properties stated in the initial situation. Consequently, the set of epistemic inclusion assertions, corresponding to the dynamic axioms whose antecedent is satisfied by $init$ in $FOE(\Sigma)$, implies the same set of properties on another state (say y_1) that is the same in each interpretation $\mathcal{I}$ of $\mathcal{W}$. That is, y_1 satisfies *at least* the same properties satisfied by x_1. Now, let x_2 be the R_{i_2}-successor of x_1. The same kind of reasoning can be applied, thus showing that there must exist a state y_2 in $\mathcal{W}$ satisfying in each world at least the same properties verified by x_2. By iteration we conclude that there exists a state y_n such that $x \in G^{\mathcal{I},\mathcal{W}}$ for all $\mathcal{I} \in \mathcal{W}$.

As for the computational aspects of reasoning about actions in the epistemic framework based on $\mathcal{ALCK}$, it turns out that, under the assumption of acyclic static axioms, the planning problem is PSPACE-complete.

Theorem 3 Let $\Sigma = \langle \Gamma_S \cup \Gamma_P \cup \Gamma_E, \{S(init)\}\rangle$, such that Γ_S is an acyclic set of $\mathcal{ALC}$ inclusion assertions. Then, the problem of establishing whether there exists a state x such that $FOE(\Sigma) \models G(x)$ is PSPACE-complete.

Proof PSPACE-hardness follows from the fact that the subsumption problem for acyclic $\mathcal{ALC}$ TBoxes, which is PSPACE-complete [11], can be reduced ($\Gamma_P \cup \Gamma_E = \emptyset$) to the problem of establishing whether there exists a state x such that $FOE(\Sigma) \models G(x)$. Membership in PSPACE is due to the fact that the maximum number of states generated in $FOE(\Sigma)$ is 2^n (where n is the number of dynamic axioms), therefore, if there exists a state x in $FOE(\Sigma)$ such that $G(x)$ holds, then such a state can be generated through a sequence of actions whose length is less or equal to 2^n. This property allows for the generation of all the states, one at a time, using a polynomial amount of space.

Therefore, from Proposition 1 it follows that, under the assumption of acyclic static axioms, the planning problem is easier (unless EXPTIME=PSPACE) in the epistemic setting than in the non-epistemic one. An informal explanation of this result may be given in the following way. In the epistemic framework we model static axioms as acyclic inclusion assertions, and we model the dynamic axioms, for which the acyclicity condition would be too restrictive, by using epistemic inclusion assertions of a special form. Specifically, in a way similar to the formalization of procedural rules, the presence of epistemic concept expressions of the form $\mathbf{K}C$ in the left-hand side of such inclusion assertions inhibits the usage of dynamic axioms for *contrapositive reasoning*, and this weakening allows for lowering the computational cost of reasoning in the epistemic framework.

Notice that the algorithm for computing $FOE(\Sigma)$ in Fig. 2 uses exponential space, because $FOE(\Sigma)$ can be used to *find* plans for any goal, not only to know whether there exists a plan. It is easy to modify the above algorithm in order to answer to the plan existence problem using polynomial space only.

5 The Mobile Robot "Tino"

Our approach to reasoning about actions has been implemented on the *Erratic* base [29]. The robot, named Tino, has been successfully tested on several real and simulated office environments.

Tino is based on a two-level architecture, which combines a reactive control mechanism with a planning system. The idea of multi-level architecture dates back to the robot Shakey [38], in which the planning system was STRIPS. However, the Erratic base allows for an effective combination of both horizontal and vertical decomposition (see [8]). In this way the system can react immediately in dynamic environments, while different representations of the environment are used by different modules.

The reactive capabilities of the robot Tino are based on a fuzzy controller [41, 30], that provides integrated routines for sonar sensor interpretation, map building, and navigation. The control problem is decomposed into small units of control, called (low-level) *behaviors*, that are distinguished in reactive ones, like avoiding obstacles, and (low-level) goal-oriented ones, like following a corridor. A *blending* mechanism is used to integrate reactive and goal-oriented behaviors, so that the robot can follow a corridor while avoiding obstacles.

As each module has its own representation of information, the exchange of information between the two layers is a critical issue. The communication between the planning system and the reactive controller is realized by a monitor, which takes care of the integration of planning and control, by both translating high-level actions into goal-oriented behaviors, and scheduling the activation of these behaviors. Indeed, high-level actions can be seen as goals to be achieved by the controller through the activation of appropriate behaviors [2]. Moreover, the monitor checks the correct execution of the behaviors, and returns to the planner a possible plan failure.

Below we describe the basic elements of the implementation of the planning component and of the plan execution component.

5.1 Plan Generation

One of the motivations underlying our proposal for reasoning about actions is the possibility of relying on a knowledge representation system based on DLs for the implementation. In particular, we have chosen CLASSIC [7], a well-known system based on DLs, to take advantage of an efficient and reliable reasoning system. However, the language for representing knowledge in CLASSIC is less expressive than the DL we have considered so far. So, we use a subset of the DL language corresponding to some of the constructs available in CLASSIC, that we write, for ease of notation, using $\sqcap$ for AND and $\forall$ for ALL. The concept expression $\exists\mathbf{K}R.\top$ has been implemented by defining a procedural hook in CLASSIC, that we call KAPPA and abbreviate with $\exists_{\mathrm{K}}$. Furthermore, we make use of the built-in instance checking mechanism to check the validity of a concept in a state, and of triggering of rules to propagate effects of actions.

Static axioms are expressed either as inclusion assertions ($\dot{\leq}$) or as concept definitions, written $\doteq$ and interpreted as necessary and sufficient conditions (see for example [9]). In this way we can write

$$A \;\dot{\leq}\; C \qquad\qquad A \doteq C$$

to define the primitive concept A as a subset of C, or equivalent to C, respectively. In both cases cycles are not allowed.

Dynamic axioms are represented as CLASSIC *rules*, denoted with $\mapsto$. Observe that they implement the epistemic sentences $\mathbf{K}C \sqsubseteq D$, hence the rule is fired only on named individuals satisfying the antecedent of the rule. Rules representing action precondition axioms and effect axioms are thus written respectively as

$$C \mapsto \exists_{\mathrm{K}} R \qquad\qquad C \mapsto \forall R.D$$

Notice that the epistemic inclusion assertion $\mathbf{K}C \sqsubseteq \forall\mathbf{K}R.\mathbf{K}D$ is correctly implemented in this setting by the CLASSIC rule $C \mapsto \forall R.D$, since in CLASSIC rules are only applied to individuals explicitly mentioned in the knowledge base.

When C is the same concept expression, we sometimes abbreviate $C \mapsto \exists_{\mathrm{K}} R$ and $C \mapsto \forall R.D$ as $C \mapsto \exists_{\mathrm{K}} R \sqcap \forall R.D$.

Finally, concept instance assertions are used to describe initial state properties. Thus

$$C(init)$$

states that C is valid in the initial state.

The goal to be achieved is expressed by means of a concept describing the properties that must hold in the final state, and the plan generation procedure tries to find a plan, that is a sequence of actions (roles), that, starting from the initial state, leads to a state in which the concept expressing the goal holds.

Let us further comment on the use of KAPPA for implementing $\mathbf{K}$. The procedural meaning of the KAPPA operator is given according to the notion of known individuals at the epistemic level, thus an individual x is instance of $(\texttt{KAPPA}\ R)$ if and only if there exists a named individual y such that $(x, y) \in R^{\mathcal{I}}$. In fact, if x is asserted to belong to $(\texttt{KAPPA}\ R)$ and there is no named R-successor of x, then a new named individual is created and added to the knowledge base as R-successor of x. Subsequently, the new individual can be used for firing rules.

The generation of plans can be obtained by slightly modifying the algorithm for computing the FOE. Indeed, it is sufficient to compute only a portion of the FOE, by terminating the computation as soon as a state satisfying the goal is generated. Then, one can build the plan, namely a term representing a sequence of actions that connect the initial state with the state in which the goal is satisfied. The implementation relies on the following assumptions (directly derived from the FOE algorithm), that are necessary for the termination of the plan generation procedure: (i) rules involving KAPPA must be applied only when no other rule can be fired; this prevents incomplete state generations, i.e. new states are generated only after the properties of existing ones have been completely derived; (ii) when a new individual z is created and all the rules able to add properties to it are fired, a search for an existing individual that is equivalent to z (i.e. described by the same concepts) is done, and if such an individual (say x) is found, $z = x$ is imposed, so that individuals are not indefinitely generated.

It is worth noting that different activations of the plan generation procedure may compute the same part of the FOE several times. Therefore, we have adapted the algorithm to reuse the already computed portion of the FOE, until a plan execution failure points out that the knowledge base needs to be updated. This modification can lead to a significant reduction of the computation time. Furthermore, when the construction of the entire FOE is feasible, a plan can be obtained in two steps. First, one generates the entire FOE following the algorithm in Fig. 2; then the plan is searched for in the FOE. In this way, the first step can be seen as a pre-processing of the knowledge base and it allows for a faster plan generation.

The complexity of the plan generation procedure above is polynomial with respect to the size of FOE(Σ), that is the number of new states (individuals) generated by the FOE algorithm, and so, in general, exponential with respect to to the size of Σ.

We observe that, under the assumption that dynamic axioms are of the form $C \mapsto \exists_{\mathrm{K}} R \sqcap \forall R.D$ and for each action R the precondition C in dynamic axioms involving R are disjoint from each other, we can make use of the CLASSIC construct FILLS instead of KAPPA for writing such axioms. The FILLS construct, that we denote as $\exists R.\{a\}$, has the following semantics:

$$(\exists R.\{a\})^{\mathcal{I}} = \{x \in \Delta \mid (x, a) \in R^{\mathcal{I}}\}$$

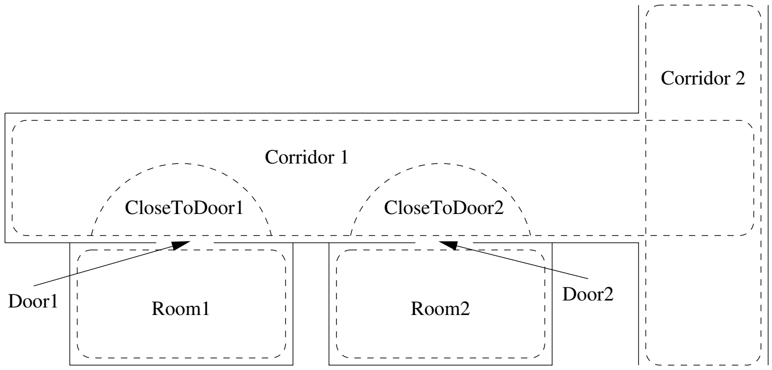

Figure 3: A simple environment

and can be intuitively interpreted as the set of individuals having a as R-successor. Note that a new individual a is created, if it does not exist, and therefore it is a known individual at the epistemic level. So dynamic axioms can be written as

$$C \mapsto \exists R.\{a\} \sqcap \forall R.D$$

In this way, computing the FOE is done in polynomial time with respect to the size of the knowledge base, because the number of individuals is at most linear in the number of rules. Hence the plan generation task is done in polynomial time too. However the restriction introduced for using `FILLS` limits the expressiveness in the representation of environments.

Example 4 Given the map shown in Fig. 3, referring to an office environment constituted by rooms, doors and corridors, we can represent the environment and Tino's capabilities of acting in it through the following knowledge base:

$$\begin{array}{lcl}
Corridor1 & \dot{\leq} & Corridor \\
Corridor2 & \dot{\leq} & Corridor \\
Room1 & \dot{\leq} & Room \\
Room2 & \dot{\leq} & Room \\
CloseToDoor1 & \dot{\leq} & Corridor1 \\
CloseToDoor2 & \dot{\leq} & Corridor1
\end{array}$$

$$\begin{array}{lcl}
Corridor1 & \mapsto & \exists_{\mathrm{K}} FollowC1ToD1 \sqcap \forall FollowC1ToD1.CloseToDoor1 \\
Corridor1 & \mapsto & \exists_{\mathrm{K}} FollowC1ToD2 \sqcap \forall FollowC1ToD2.CloseToDoor2 \\
Corridor1 & \mapsto & \exists_{\mathrm{K}} FollowC1ToC2 \sqcap \forall FollowC1ToC2.Corridor2 \\
CloseToDoor1 & \mapsto & \exists_{\mathrm{K}} EnterD1 \sqcap \forall EnterD1.Room1 \\
CloseToDoor2 & \mapsto & \exists_{\mathrm{K}} EnterD2 \sqcap \forall EnterD2.Room2 \\
Room1 & \mapsto & \exists_{\mathrm{K}} ExitD1 \sqcap \forall ExitD1.CloseToDoor1 \\
Room2 & \mapsto & \exists_{\mathrm{K}} ExitD2 \sqcap \forall ExitD2.CloseToDoor2
\end{array}$$

$Corridor1(init)$

The FOE corresponding to the above knowledge base is given in Fig. 4. Observe that, since in this case the requirement for using `FILLS` is satisfied, we compute the FOE in polynomial time.

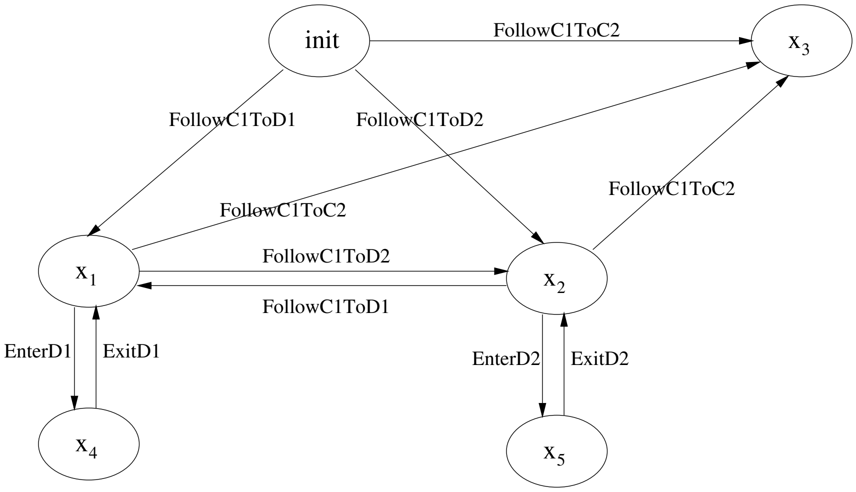

Figure 4: The FOE

In the above example, the concept *CloseToDoor1* is defined as a specialization of *Corridor1*, that is when the robot is close to the first door, it is also in the first corridor. Now the dynamic axiom involving the *ExitD1* action has just *CloseToDoor1* as effect while in the FOE (see Fig. 4) from state x_1 (successor state with respect to *ExitD1*), the actions *FollowC1ToC2* and *FollowC1ToD2* (whose precondition is *Corridor1*) are allowed. In other words, static axioms can produce many edges of the graph with a single dynamic axiom.

The ability to define a taxonomic representation of the environment not only provides for a more compact way of specifying the knowledge about the problem domain, but also allows one to have a more flexible and easy-to-modify representation. In particular, we have taken advantage of these features both in modeling different environments and in implementing a module that takes as input a topological representation of the map and generates a CLASSIC knowledge base.

5.2 Dynamic execution of plans

The dynamic execution of plans is achieved by a monitor, that activates the behaviors associated with high-level actions, keeps track of the current state of the robot on the FOE and checks the correct execution of these behaviors replying to the planner if a plan fails.

Fig. 5 shows the execution of a simple plan (*FollowC1ToD2; EnterD2*) to reach *Room2* in the environment described in Example 5.1. Notice the interaction of goal oriented and reactive behaviors when the robot has to avoid the unknown obstacle while following the corridor.

Example 5 We can describe the fact that the robot can enter in a room only if it is close to an open door by replacing the dynamic axioms involving *Enter* actions in Example 5.1 with

$CloseToOpenDoor1 \mapsto \exists_{\mathrm{K}} EnterD1 \sqcap \forall EnterD1.Room1$
$CloseToOpenDoor2 \mapsto \exists_{\mathrm{K}} EnterD2 \sqcap \forall EnterD2.Room2$

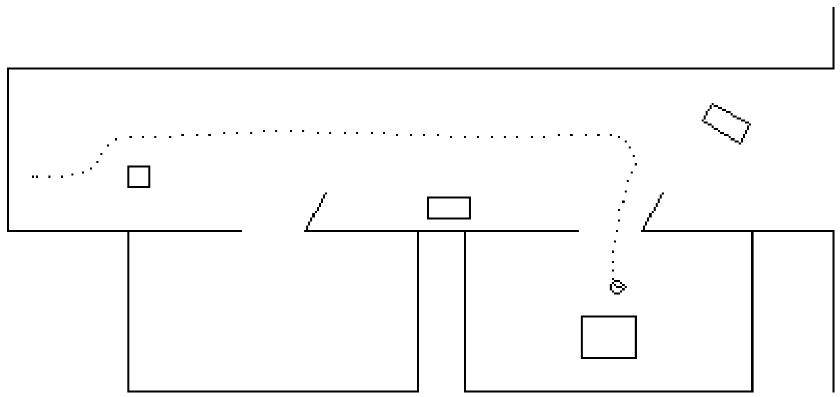

Figure 5: Plan execution for reaching Room2

and by adding the following axioms:

$$CloseToOpenDoor1 \doteq CloseToDoor1 \sqcap OpenDoor1$$
$$CloseToOpenDoor2 \doteq CloseToDoor2 \sqcap OpenDoor2$$

$$OpenDoor1 \mapsto \forall FollowC1ToD1.OpenDoor1$$
$$OpenDoor1 \mapsto \forall FollowC1ToD2.OpenDoor1$$
$$\ldots$$
$$OpenDoor2 \mapsto \forall FollowC1ToD1.OpenDoor2$$
$$\ldots$$

$$OpenDoor1 \sqcap OpenDoor2 \sqcap Corridor1(init)$$

Notice the use of axioms denoting the persistence of properties through actions (e.g. *OpenDoor1* is not changed by executing action *FollowC1ToD1*). In this way properties concerning the status of the doors are propagated through all the states in the action graph, so, when the robot reaches the state *CloseToDoor1*, it can deduce that *CloseToOpenDoor1* is valid too, and it can enter the door.

The monitor is also responsible for plan failure management. When an action is not successfully executed, the system can use information about possible failures to update the knowledge base and generate a new plan.

For example, if the action *EnterD1* fails, we can assume that the door 1 is closed, so the system has to update the current state of the robot after the failure (i.e. it returns a new current state in which *OpenDoor1* is not valid), and a new planning task is performed. In Fig. 6 the execution of a plan, whose goal is to enter in any room (specified by the concept *Room*), is shown. Note that the robot fails to enter the first door, and after a replanning task, the second one is successfully entered.

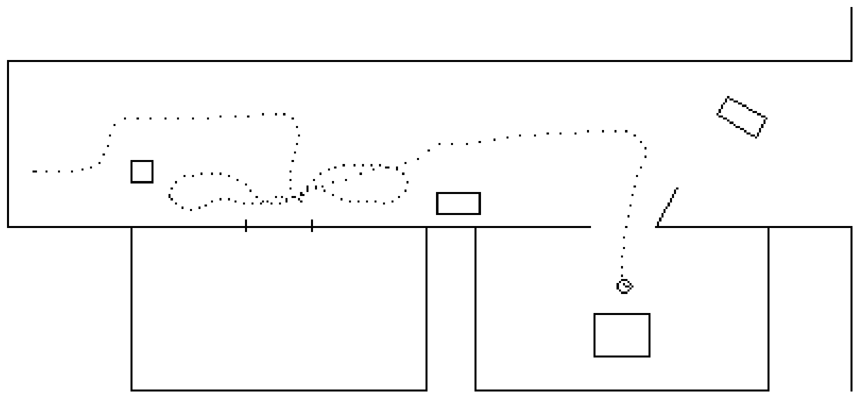

Figure 6: Plan execution for reaching any room

6 Related Work

Autonomous navigation for mobile robots has been extensively studied from many different perspectives. A large debate from an architectural viewpoint was carried out in the last years (deliberative planning vs. reactivity), and many different planning techniques and systems were proposed.

From the architectural point of view, in [8] a layered architecture for controlling mobile robots is proposed. While previous approaches were mainly concerned with automatic plan generation and relied on complete and correct knowledge about the world, the focus of the layered architecture was in the ability to quickly react to unexpected situations. Many reactive systems for controlling mobile robots have been proposed [1, 22, 27, 46], in which the task of plan generation is either not addressed or considered as the task of specifying reactions for each possible situation.

The integration of deliberative planning and reactive capabilities has been usually addressed by defining different levels for planning and control. In many cases [26, 47, 41] world model and plan representation are shared for coordinating the subsystems. Instead, in [24] a heterogeneous and asynchronous architecture is presented in order to combine classical AI techniques with reactive control mechanisms. Our approach can be considered as heterogeneous, since it separates the planning component from the underlying control mechanisms. However, we maintain, through the monitor, a strict correspondence between the representation of the world and the plan in the two levels.

On the other hand, a significant application of high-level control of mobile robots can be found in Shakey the robot [38], in which the STRIPS representation system was used for planning, and in the robot programming language based on Situation Calculus [32], where a plan is specified through a program. Our proposal, by relying on Description Logics, provides for an expressive formalism intermediate between propositional STRIPS and Situation Calculus. Moreover, such a formalism is actually implemented on an real mobile robot through a knowledge representation tool, which enables for the automatic generation of plans.

Several studies propose DLs for the development of planning systems (among them [16, 3, 28]). In these proposals the language of DLs is extended with specific constructs that allow actions to be represented as concepts. The planning system can thus reason about plans, by exploiting subsumption in DLs. Our proposal takes a different perspective, derived from the correspondence with PDLs, where actions are represented as roles, and properties of states as concepts. In our case, plans are generated through a combination of the propagation mechanism for the procedural rules and taxonomic reasoning for checking the static properties of states.

Finally, the ability of representing and reasoning about the epistemic state of the agent is convenient to deal with conditional plans and sensing (or knowledge-producing) actions [33]. Indeed, there are several logic-based approaches to reasoning about actions which address this topic (see e.g. [34, 25, 35]). In particular, the possibility of expressing and reasoning about the epistemic state of the agent has been studied in the framework of the Situation Calculus in [43], which shows the possibility of representing a knowledge modality by using the language of the Situation Calculus. In principle this allows for representing effect axioms in terms of procedural rules in a way analogous to what we have shown in Section 4. However, the issue of implementation and the computational aspects of the corresponding planning task have not yet been addressed.

7 Conclusions

The goal of our work was to devise a principled realization of reasoning agents in the realm of mobile robots. In the paper we have presented a formal setting for reasoning about actions and its implementation on the *Erratic* base, integrating reacting behavior with action planning and execution.

We believe that the results of our work are twofold. From the standpoint of the representation of dynamic systems, the origin of our work is in the formal correspondence between Propositional Dynamic Logics and Description Logics. This has led us not only to a semantically justified implementation, but also to the adaptation of the formal setting in new and interesting directions. In particular, we have shown that the epistemic representation of actions simplifies the planning task and (under some restrictions) allows for the implementation of the theoretical framework for reasoning about actions in an efficient knowledge representation system (CLASSIC).

From the standpoint of implementation, the choice of a knowledge representation system, together with the associated methodology for representing the dynamic environment and reasoning about actions, has led to a very flexible implementation of the planning component, that can be easily adapted to new environments. To this end, the possibility of structuring the representation of environments using the features of a DL representation language plays a crucial role. Moreover, our implementation has been successful in a real environment because of the underlying system architecture, which provides a powerful and accessible set of functionalities. By exploiting such functionalities we obtained an effective integration of the reasoning capabilities of the agent.

The framework developed so far is mainly concerned with the position of the robot and its moving abilities. We are currently working at several extensions of the robot capabilities, and, consequently, of the framework for reasoning about actions, that will enable Tino to address more complex scenarios.

First, it is possible in this framework to provide the robot with the ability of reasoning about its reactive abilities, in order to plan different kinds of reactive behaviors in different situations. For example, it could be useful to provide the robot with different kinds of obstacle avoidance behaviors: the robot could activate at the deliberative level the appropriate one, according to the kind of obstacle encountered.

Second, the notion of epistemic state of the agent can be exploited, since it can be used to address several issues arising in complex dynamic domains. In particular, we are currently focusing on the following aspects:

1. *Frame problem.* It turns out [20] that a slight extension of $\mathcal{ALCK}$ allows for the representation of default rules, thus allowing for the formalization of the notion of default persistence of knowledge, realized through the use of such rules. This property, together with the notion of default persistence of ignorance encoded in the semantics of $\mathcal{ALCK}$, allows, in principle, for a formalization of the common-sense law of inertia, through a "small" number of epistemic axioms (formalizing default rules).

2. *Sensing.* The usage of the epistemic operator of $\mathcal{ALCK}$ can be exploited in our framework for reasoning about actions, in order to formalize sensing (or knowledge-producing) actions. Indeed, the possibility of representing the epistemic state of the agent allows for a simple treatment of actions whose execution only changes the knowledge of the agent without affecting the state of the world. The first results in this direction are reported in [13].

3. *Plan constraints.* Our framework also allows in principle for addressing the problem of specifying *temporally extended goals* [6] and *dynamic constraints on plans* [5], which have been shown effective for speeding up the planning process.

4. *Multi-agents.* The epistemic nature of our framework makes it feasible to address multi-agent scenarios where an agent can reason on other agents' knowledge (see [21]). In this respect, one can take advantage of the studies on multi-modal DLs [4].

Acknowledgments

The work has been supported by Italian MURST "Pianificazione di azioni per robot mobili". Riccardo Rosati acknowledges Kurt Konolige and the AI Center of SRI International, Menlo Park, for the support provided during his visiting period.

References

[1] P. E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 1987.

[2] P. E. Agre and D. Chapman. What are plans for? *Robotics and Autonomous Systems*, 6:17–34, 1990.

[3] A. Artale and E. Franconi. A computational account for a description logic of time and action. *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, 1994.

[4] F. Baader and A. Laux. Terminological logics with modal operator. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[5] F. Bacchus and F. Kabanza. Using temporal logic to control search in a forward chaining planner. In *Proocedings of the Third European Workshop on Planning*, 1995.

[6] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1215–1222, 1996.

[7] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989.

[8] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), 1986.

[9] M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf. Terminological systems revisited: Terminology = schema + views. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 199–204, Seattle, USA, 1994.

[10] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[11] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 303–307. John Wiley & Sons, 1996.

[12] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Moving a robot: the KR&R approach at work. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 198–209, 1996.

[13] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Planning with sensing for a mobile robot. In *Proceedings of the Fourth European Conference on Planning (ECP-97)*, 1997.

[14] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[15] G. De Giacomo and M. Lenzerini. PDL-based framework for reasoning about actions. In *Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA-95)*, number 992 in Lecture Notes In Artificial Intelligence, pages 103–114. Springer-Verlag, 1995.

[16] P. Devambu and D. Litman. Plan-based terminological reasoning. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 128–138. Morgan Kaufmann, Los Altos, 1991.

[17] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. Adding epistemic operators to concept languages. In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 342–353. Morgan Kaufmann, Los Altos, 1992.

[18] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. Queries, rules and definitions. In *Foundations of Knowledge Representation and Reasoning*. Springer-Verlag, 1994.

[19] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[20] F. M. Donini, D. Nardi, and R. Rosati. Autoepistemic description logics. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 136–141, 1997.

[21] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[22] R. J. Firby. An investigation into reactive planning in complex domains. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 1987.

[23] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[24] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 1992.

[25] R. Goldman, M. Boddy, and L. Pryor. Planning with observations and knowledge. In *AAAI-97 workshop on theories of action, planning and control*, 1996.

[26] S. Hanks and R. J. Firby. Issues and architectures for planning and execution. In *Proc. of Workshop on Innovative Approaches to Planning, Scheduling, and Control*, 1990.

[27] L. Kaelbling and S. Rosenschein. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6:35–48, 1990.

[28] J. Koehler. An application of terminological logics to case-based reasoning. *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, 1994.

[29] K. Konolige. Erratic competes with the big boys. *AAAI Magazine*, Summer:61–67, 1995.

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

[31] D. Kozen and J. Tiuryn. Logics of programs. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science – Formal Models and Semantics*, pages 789–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[32] Y. Lesperance, H. Levesque, F. Lin, D. Marcu, R. Reiter, and R. Scherl. A logical approach to high-level robt programing. In *AAAI FAll Symposium on Control of the Physical World by Intelligent Systems*, 1994.

[33] H. J. Levesque. What is planning in presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1139–1149. AAAI Press/The MIT Press, 1996.

[34] F. Lin and Y. Shoham. On non-forgetting and minimal learning. In *Proc. of the 1993 Int. Coll. on Cognitive Science*, 1993.

[35] J. Lobo, G. Mendez, and S. Taylor. Adding knowledge to the action description language A. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 454–459, 1997.

[36] D. McAllester, 1991. Unpublished Manuscript.

[37] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[38] N. J. Nilsson. Shakey the robot. Technical Report 323, SRI Artificial Intelligence Center, 1984.

[39] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

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

[41] A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.

[42] E. Sandewall and Y. Shoham. Non-monotonic temporal reasoning. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 439–498. Oxford Science Publications, 1995.

[43] R. Scherl and H. J. Levesque. The frame problem and knowledge producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 689–695, 1993.

[44] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, Australia, 1991.

[45] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48(1):1–26, 1991.

[46] M. Schoppers. Universal plan for reactive robots in unpredictable environments. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 1987.

[47] R. Simmons. An architecture for coordinating planning, sensing and action. *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 1992.

[48] W. A. Woods and J. G. Schmolze. The KL-ONE family. Technical Report TR-20-90, Aiken Computation Laboratory, Harvard University, Cambridge, MA, 1990. To appear in *Computers & Mathematics with Applications*.

Received 1 November 1997