Moving a Robot: The KR&R Approach at Work

Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
email: {degiacomo,iocchi,nardi,rosati}@dis.uniroma1.it

Abstract

The paper describes an approach to reasoning about actions and plan generation within the framework of description logics. From an epistemological viewpoint, our approach is based on the formalization of actions given by dynamic logics, but we exploit their correspondence with description logics to turn the formalization into an actual implementation. In particular, we are able to carefully weaken the logical inference process, thus making the reasoning of the robot computationally feasible. From a practical viewpoint, we use a general purpose knowledge representation environment 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. The robot's reactive capabilities allow it to execute such plans in the real world.

1 INTRODUCTION

We present one attempt to reconcile the theoretical work in knowledge representation with the implementation of real systems. The realm we address is that of mobile robots, which has always been considered central to Artificial Intelligence. Recent work in this field (see for example [Brooks,1986]) has shown that, in order to enable a mobile robot to cope with the uncertainties and dynamics of real environments, some kind of reactive behavior is necessary. 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 the research in mobile robots and planning systems (see for example [Saffiotti *et al.*,1995; Kaelbling and Rosenschein,1995; Simmons,1992; Gat,1992]).

In the present work, we provide a framework for reasoning about actions and discuss its implementation, through a knowledge-based system, on a robot with reactive capabilities. Our approach falls in the research stream of logic-based approaches for reasoning about actions (see [Lesperance *et al.*,1994]), however it has been developed as a balance between theoretical and practical considerations. Specifically, the basis of our proposal for reasoning about actions is provided by Propositional Dynamic Logics (PDLs), following the work of [Rosenschein,1981; De Giacomo and Lenzerini,1995b]. In this setting PDLs formulae denote properties of states, and actions (also called programs) denote state transitions from one state to another. The dynamic system itself is described by means of axioms. Two kinds of axioms are introduced, "static axioms", that describe background knowledge, and "dynamic axioms", that describe how the situation changes when an action is performed. As in the deductive-planning tradition, a plan can be generated by finding a constructive existence proof for the state where the desired goal is satisfied. In a PDL setting a plan consists of a sequence of transitions, which leads to a state satisfying the goal.

The novel and fundamental step towards the implementation has been to rely on the tight correspondence that exists between PDLs and Description Logics (DLs) [Schild,1991; De Giacomo and Lenzerini,1994]. By exploiting this correspondence we have been able both to develop an interesting theoretical framework for reasoning about actions and to obtain an implementation that uses a knowledge representation system based on DLs.

The work on efficient reasoning methods in DLs shows that the typical form of dynamic axioms is problematic wrt efficiency (such axioms are "cyclic" in the DLs terminology). 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 [Donini *et al.*,1994] 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 computationally feasible and semantically justified approach to deductive planning.

Indeed, there are several studies that propose to use DLs as a basis for the development of planning systems (among them [Artale and Franconi,1994; Borgida,1992; Koehler,1994]). These works extend the DLs language 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.

We have built an implementation on top of the mobile robot *Erratic*, equipped with wheels and sonar, which has the capability of integrating action execution and reactive behavior [Konolige,1995]. The knowledge representation system used in the implementation is CLASSIC [Borgida *et al.*,1989], a well-known, general-purpose knowledge representation system based on DLs. One interesting feature of the implementation is that it relies on the reasoning tools provided by such a system, although in this way the formalization is restricted to the subset of theories expressible in the CLASSIC representation language. We named our mobile robot "Tino" and demonstrated it at the 1995 Description Logic Workshop.

The paper is organized as follows. In Section 2, we present the general framework for the representation of dynamic systems we have adopted. In Section 3 we introduce Epistemic DLs, and in Section 4 we address our specific way of representing and reasoning about actions in such a formalism. Finally, in Section 5, we describe the mobile robot "Tino", which includes the implementation in CLASSIC of the planning component and a module for exchanging information between high-level planning and the software implementing the reactive capabilities of the robot.

2 REASONING ABOUT ACTIONS: THE GENERAL FRAMEWORK

In this section we present the general framework for representing dynamic systems on which our work is based. Such a framework is essentially that of PDLs [Rosenschein,1981; De Giacomo and Lenzerini,1995a].

Dynamic systems are typically modeled in terms of state evolutions caused by actions. A *state* represents a situation the system can be in, and is characterized by a set of properties which forms a *complete* description (wrt some logic/language) of the represented situation. *Actions*, which are typically considered deterministic, cause state transitions, making the system evolve from the current state to the next one.

In principle we could represent the *behavior* of a system, i.e. all its possible evolutions, as a *transition graph*, where each node denotes a state, and is labeled with the properties that characterize the state, 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 build its transition graph, while in general one has only partial knowledge of such behavior. In deductive planning this knowledge is phrased in *axioms* of some logic (e.g. PDLs [Rosenschein,1981] or Situation Calculus [Reiter,1993]). These axioms select a subset of all possible transition graphs. All the selected graphs are similar, since they all satisfy the same axioms, but yet different wrt those properties not imposed by the axioms. Hence one has to concentrate on those properties that are true in all the selected graphs, i.e. those properties that are *logically implied* by the axioms.

Following [Rosenschein,1981] two kinds of axioms are distinguished:

- *Static axioms*, which are used for representing background knowledge that is invariant with respect to the execution of actions. In other words, static axioms hold in any state and do not depend on actions.
- *Dynamic axioms*, which are introduced to represent the changes actions bring about, and have the form[1]:
$$C \Rightarrow \langle R \rangle \mathtt{tt} \wedge [R]D$$
where R is an action, C represents the *preconditions* that must hold in a state, in order for the action R to be executable; D denotes the *postconditions* that are true in the state resulting from the execution of R in a state where preconditions C hold. Multiple axioms per action are allowed.

In deductive planning one is typically 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 (here we phrase it in PDLs):

$$\Gamma \models S \Rightarrow \langle \alpha^* \rangle G \qquad (1)$$

where: (i) Γ is the set of both static and dynamic axioms representing the (partial) knowledge about the

[1] Actually in [Rosenschein,1981] the form of the dynamic axioms is $C \Rightarrow [R]D$, and for each action R the axiom $\langle R \rangle \mathtt{tt}$ is assumed to be valid.

DLs		PDLs	
atomic concept	A	atomic proposition	A
top	$\top$	true	`tt`
bottom	$\bot$	false	`ff`
conjunction	$C \sqcap D$	conjunction	$C \wedge D$
disjunction	$C \sqcup D$	disjunction	$C \vee D$
negation	$\neg C$	negation	$\neg C$
existential quantification	$\exists R.C$	diamond ("some runs ...")	$\langle R \rangle C$
universal quantification	$\forall R.C$	box ("all runs ...")	$[R]C$
inclusion assertion	$C \sqsubseteq D$	valid implication (axiom)	$C \Rightarrow D$
instance assertion	$C(\alpha) \mid R(\alpha_1, \alpha_2)$	—	—

Figure 1: Correspondence between DLs and PDLs.

system; (ii) S is a formula representing the (partial) knowledge about the initial situation (state); (iii) G is a formula representing the goal, which is, in fact, a (partial) description of the final state one wants to reach; (iv) $\langle \alpha^* \rangle G$ (where $\langle \alpha^* \rangle G$ stands for any formula of the form $\langle R_1 \rangle \langle R_2 \rangle \ldots \langle R_n \rangle G$ with $n \geq 0$ and R_i any action) expresses the existence of a finite sequence of actions leading to a state where G is satisfied.

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 a 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 incurs a high computational cost (typically PDLs are EXPTIME-complete [Kozen and Tiuryn,1990]).

Therefore, we make use of the correspondence between PDLs and Description Logics (DLs) to take advantage of the extensive studies of the computational aspects of reasoning in DLs, and to exploit specific techniques developed in DLs to lower the cost of reasoning. We present our proposal using the notation of DLs, in order to make it easier to relate our proposal both to previous research in DLs, whose results are exploited here, and to the actual implementation in CLASSIC.

3 EPISTEMIC DESCRIPTION LOGICS

In this section we introduce description logics with an epistemic operator, which constitute the technical background of our proposal. We focus on a well-known DL, $\mathcal{ALC}$, and its epistemic extension, $\mathcal{ALCK}$, obtained by adding a modal operator interpreted in terms of minimal knowledge as in [Donini *et al.*,1992; 1994; 1995].

In Fig. 1 we present the constructs of the DL $\mathcal{ALC}$ and the correspondence with PDLs. Such correspondence, first pointed in [Schild,1991], is based on the similarity between the interpretation structures of the two kinds of logics. At the extensional level, states in PDLs correspond to individuals (members of the domain of interpretation) in DLs, whereas state transitions correspond to links between two individuals. At the intensional level, propositions correspond to concepts, and actions correspond to roles. The correspondence is realized through a (one-to-one and onto) mapping from PDLs formulae to DLs concepts, and from PDLs actions to DLs roles. For a detailed presentation of such a mapping and more generally of the correspondence we refer to [Schild,1991; De Giacomo and Lenzerini,1994]. For our purposes it suffices to consider DLs concepts and roles as syntactic variants of PDLs formulae and actions respectively.

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

Figure 2: Semantics of $\mathcal{ALC}$

In the following we shall refer to an *$\mathcal{ALC}$-interpretation* $\mathcal{I}$ as a function mapping each concept expression into a subset of some abstract interpretation domain Δ and each role expression into a subset of $\Delta \times \Delta$, such that the equations of Fig. 2 are satisfied.

The basic reasoning service for DLs is subsumption: C is subsumed by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every $\mathcal{I}$. In

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

Figure 3: Semantics of $\mathcal{ALCK}$

other words subsumption allows to establish a hierarchy among concept descriptions, which can be used to reason on a specific problem instance. The complexity of subsumption in DLs has been carefully studied for different logics obtained by admitting different sets of constructs.

$\mathcal{ALCK}$ is an extension of $\mathcal{ALC}$ with an epistemic operator interpreted as knowledge. More precisely the $\mathcal{ALCK}$ abstract syntax is as follows (C, D denote concepts, R denotes a role, A denotes a primitive concept, and P a primitive role):

$$
\begin{aligned}
C, D \quad &::= \quad A \mid \top \mid \bot \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \\
&\qquad \forall R.C \mid \exists R.C \mid \mathbf{K}C \\
R \quad &::= \quad P \mid \mathbf{K}P
\end{aligned}
$$

Non-epistemic concepts and roles are given essentially the standard semantics of DLs, conversely epistemic sentences 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.

The semantics is based on the Common Domain Assumption: Every interpretation is defined over the same, fixed, countable-infinite domain of individuals Δ.

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 $A^{\mathcal{I},\mathcal{W}} \subseteq \Delta$, $P^{\mathcal{I},\mathcal{W}} \subseteq \Delta \times \Delta$ and the equations in Fig. 3 are satisfied.

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 $\mathcal{J} \in \mathcal{W}$. In other worlds, an individual is known to be an instance of a concept if it belongs to the concept interpretation of every possible world. 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 $\mathcal{J} \in \mathcal{W}$.

DLs are typically used for representing the knowledge about a problem domain by providing mechanisms both for introducing concept definitions and for specifying information about individuals. Accordingly, an $\mathcal{ALCK}$ *knowledge base* Ψ is defined as a pair $\Psi = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$, called the *TBox*, is a finite set of inclusion statements of the form $C \sqsubseteq D$, with $C, D \in \mathcal{ALCK}$, and $\mathcal{A}$, called the *ABox*, is a finite set of membership assertions of the form $C(a)$ or $R(a, b)$, where $C, R \in \mathcal{ALCK}$ and a, b are *names* of individuals. We assume that different names denote different individuals, hence, we generally do not distinguish between individuals and their names.

The truth of inclusion statement is defined in terms of set inclusion: $C \sqsubseteq D$ is satisfied iff $C^{\mathcal{J},\mathcal{W}} \subseteq D^{\mathcal{J},\mathcal{W}}$. Assertions are interpreted in terms of set membership: $C(a)$ is satisfied iff $a \in C^{\mathcal{J},\mathcal{W}}$ and $R(a, b)$ is satisfied iff $(a, b) \in R^{\mathcal{J},\mathcal{W}}$. A *model* for an $\mathcal{ALCK}$-knowledge base Ψ is a set of $\mathcal{ALC}$-interpretations $\mathcal{W}$ such that for each interpretation $\mathcal{I} \in \mathcal{W}$, every sentence (inclusion or membership assertion) of Ψ is true in the $\mathcal{ALCK}$-interpretation $(\mathcal{I}, \mathcal{W})$.

The final step of the construction consists of defining a preference semantics on universal Kripke structures, which allows one to select only those model 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 Ψ is a model for Ψ 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 Ψ. Ψ is *satisfiable* if there exists a preferred model for Ψ, *unsatisfiable* otherwise. Ψ logically implies an (inclusion or membership) assertion σ, written $\Psi \models \sigma$, if σ is true in every preferred model for Ψ.

Using the epistemic operator, it is possible to formalize in $\mathcal{ALCK}$ several interesting features provided by frame systems, based on DLs [Donini *et al.*,1994]. 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 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". Therefore they can be viewed as implications for which the contrapositive does not hold. A procedural rule $C \mapsto D$ can be formalized in $\mathcal{ALCK}$ by the epistemic sentence

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

Notice that procedural rules can be regarded as a weak form of concept definitions.

A 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 the entailment problem can be solved by constructing a knowledge base, called *first-order extension* [Donini *et al.*,1992; 1994]. Informally, the first-order extension is incrementally built by applying the following procedure: For each individual i explicitly mentioned in the ABox of the knowledge base and for each rule $C \mapsto D$, if $C(i)$ is a consequence of the knowledge base, then add $D(i)$ to the knowledge base. The first-order extension thus constructed can be used for answering queries in place of the epistemic knowledge base.

4 REASONING ABOUT ACTIONS: OUR PROPOSAL

In this section we present our framework for representing dynamic systems and reasoning about them. We first introduce the representation and, subsequently, address the reasoning method.

4.1 REPRESENTATION

Let us now describe how we use epistemic DLs to formalize dynamic systems. As we want to tackle the computational cost of reasoning, two aspects must be carefully considered:

1. The expressivity of the language, i.e. the set of constructs allowed;
2. The form of the axioms, i.e. the form of the inclusion assertions.

The research in DLs has shown that there is a trade-off between expressivity and complexity of reasoning, and has devised a number of languages for which reasoning without inclusion assertions is polynomial. However, adding general inclusions of the form $C \sqsubseteq D$ makes reasoning EXPTIME-hard even for a simple language as $\mathcal{FL}_0$ [McAllester,1991], which contains only intersection $\sqcap$ and universal quantification $\forall$. Hence, restrictions on the form of the inclusions are normally considered. In particular, cycles (recursive inclusions) [Nebel,1991; Buchheit *et al.*,1993; 1994; Calvanese,1996] are especially problematic from the computational point of view, and typically are not allowed.

Taking into account the above considerations, we model static axioms as acyclic inclusion assertions, and we model the dynamic axioms, for which the acyclicity condition would be too restrictive, by making a special use of procedural rules. In this way, the dynamic axioms *cannot be used* in the reverse direction for *contrapositive reasoning*, and this weakening allows for lowering the computational cost of reasoning in our formalism.

Specifically, in our ontology, an agent is, at any given point, in a certain *state*, represented by an *individual* in the domain of interpretation. *Properties of states* are represented as *concepts* of DLs. That is, a concept denotes a property that may hold in a state. *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 useful for structuring properties of states, and *action-roles*, which are functional roles that denote actions and are used in a special way.

The behavior of the agent is described by means of both static axioms and dynamic axioms. We formalize static axioms as acyclic inclusion assertions, not involving action-roles. Whereas, by exploiting the epistemic interpretation of procedural rules [Donini *et al.*,1994; 1995], we formalize dynamic axioms through epistemic sentences of the form:

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \sqcap \forall R.D \tag{2}$$

which can be intuitively interpreted as: For all possible interpretations, if a state (individual) x is an instance of C in all possible interpretations, then there exists a state y which is the (unique) R-successor of x in all possible interpretations, and y is an instance of D. In other words, for all possible transition graphs, if a state x satisfies the property C in all possible transition graphs, then there exists a state y which is the R-successor of x in all possible transition graphs, and y satisfies D. From now on we refer to states whenever the individuals are interpreted as states.

Notably, by using dynamic axioms of this special form, we recover the ability of representing the behavior of the system by means of a *single* graph – an ability that is generally lost when using other forms of dynamic axioms. Such a graph, which we may call *partial transition graph*, summarizes the common part of all transition graphs that, by our (partial) knowledge about the dynamic system, are considered possible. The partial transition graph gives us a description of a transition graph which is partial, in the sense that:

1. Certain states and transitions may be missing;
2. The properties of the states in the graph may be only partially specified.

Given an initial state satisfying certain properties, a plan exists for a specified goal if there exists a finite sequence of actions that, from the initial state, leads to a state satisfying the goal, regardless of which of the possible interpretations corresponds to the actual world. This condition is expressed by a logical implication similar to (1), namely:

$$\langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle \models ((\exists \mathbf{K}\alpha)^*.\mathbf{K}G)(init) \qquad (3)$$

where: (i) Γ_S and Γ_D respectively indicate the sets of static axioms and dynamic axioms; (ii) *init* names an individual representing the initial state, and S is a concept describing our knowledge about such an initial state; (iii) $(\exists \mathbf{K}\alpha)^*.\mathbf{K}G$ 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). Intuitively, condition (3) checks for the existence of a state of the partial transition graph, reachable from the initial one, in which the goal is satisfied.

Observe that condition (3) holds iff for each preferred model $\mathcal{W}$ for $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle$, there exists a state $x \in \Delta$ such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$. Indeed, by 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.

4.2 REASONING

Let us now turn our attention to the problem of computing the entailment (3). We point out that in general the $\mathcal{ALCK}$-knowledge base Σ has many preferred models, which are distinguishable even up to renaming of individuals. Nevertheless, due to the special form of the epistemic sentences corresponding to the dynamic axioms in Σ, we can build the so-called *first-order extension* (FOE) [Donini *et al.*,1994] of the knowledge base $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle$, which consists of the knowledge base $\langle \Gamma_S, \{S(init)\}\rangle$ augmented by the assertions which are consequences (up to renaming of individuals) of the epistemic sentences describing the dynamic axioms. The FOE of Σ provides a unique characterization of the knowledge that is shared by all the preferred models of Σ. In fact, the notion of FOE formalizes the concept of partial transition graph introduced above, which describes all common properties of the possible behaviors of the system.

The FOE of Σ, written $FOE(\Sigma)$, is computed by the algorithm shown in Fig. 4, in which

$POST(\Sigma, R, s) =$

$$\{D_i \mid (\mathbf{K}C_i \sqsubseteq \exists \mathbf{K}R.\top \sqcap \forall R.D) \in \Sigma \wedge \Sigma \models C_i(s)\}$$

denotes the effect of the application of the rules of Σ involving the action-role R to the state s, namely the set of postconditions (concepts) of the rules which are triggered by s, and

$$CONCEPTS(\Sigma, i) = \{D \mid \Sigma \models D(i)\}$$

denotes the set of concepts verified by the state i in Σ.

Informally, the algorithm, starting from the initial state *init*, applies to each named state the rules in the set Γ_D which are triggered by such a state. A new state is thus generated, unless a state with the same properties had already been created. In this way the effect of the rules is computed, obtaining a sort of "completion" of the knowledge base.

It is easy to see that 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 renaming of states. Moreover, it is easy to see that 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 postconditions of the rules, i.e. 2^n, where n is the number of rules in Σ. 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 the postconditions of the rules in Γ_D, $\langle \Gamma_S, ABOX\rangle \models C(l)$ 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 rules (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 a 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, wrt the entailment problem (3), the first-order extension of Σ represents the information which must hold in *any* preferred model for Σ.

Theorem 4.1 *There exists a state x such that*

$$FOE(\Sigma) \models G(x) \qquad (4)$$

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

Sketch of the proof. If-part. Suppose that for each model $\mathcal{W}$ for Σ there exists a state x such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$. Now, it is easy to see that there exists a preferred model $\mathcal{W}'$ for Σ such that for each state s in $FOE(\Sigma)$ there exists a state s'

ALGORITHM FOE
INPUT: $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle$
OUTPUT: $FOE(\Sigma)$
begin
  STATES = $\{init\}$;
  ALL-STATES = $\{init\}$;
  $ABOX = \{S(init)\}$;
  repeat
    s = choose(STATES);
    for each action-role R **do**
    begin
      s' = NEW state name;
      $ABOX' = ABOX \cup \{R(s,s')\} \cup \{D_i(s') | D_i \in POST(\langle \Gamma_S \cup \Gamma_D, ABOX\rangle, R, s)\}$;
      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 4: Algorithm computing $FOE(\Sigma)$

in $\mathcal{W}$ isomorphic to s, that is, for every concept C, $FOE(\Sigma) \models C(s)$ iff $s' \in C^{\mathcal{J},\mathcal{W}}$. 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 sentences, 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 $\mathcal{J}$ 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 $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$, which concludes the proof. □

By the above property, we can solve the planning problem (3) by verifying whether there is an x in $FOE(\Sigma)$ that satisfies the goal G.

4.3 COMPLEXITY

As for the computational aspects of reasoning about actions in the epistemic framework based on $\mathcal{ALCK}$, it turns out that the planning problem is PSPACE-complete which is a direct consequence of Theorem 4.1 and of the following property.

Theorem 4.2 *The problem of establishing whether there exists a state x such that $FOE(\Sigma) \models G(x)$ is PSPACE-complete.*

Sketch of the proof. PSPACE-hardness follows from the fact that the subsumption problem for acyclic $\mathcal{ALC}$ TBoxes, which is PSPACE-complete [Calvanese,1996], can be reduced ($\Gamma_D = \emptyset$) 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. □

Notice that the algorithm for computing $FOE(\Sigma)$ uses exponential space, because $FOE(\Sigma)$ can be used to

find a plan, 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.

Let us now compare our formalization with the one we get leaving out the epistemic operator (i.e. Rosenschein's formalization), hence using $\mathcal{ALC}$ instead of $\mathcal{ALCK}$. The difference between the two formalizations lies in the different representation of the dynamic axioms Γ_D, which are formalized through epistemic sentences in $\mathcal{ALCK}$ of the form (2), whereas they are expressed by ordinary axioms (implications) in $\mathcal{ALC}$, of the following form:

$$C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

corresponding to Rosenschein's dynamic axioms

$$C \Rightarrow \langle R \rangle \mathtt{tt} \wedge [R]D$$

The planning problem (1) is expressed in such a setting by the following entailment problem:

$$\langle \Gamma_S \cup \Gamma_D, \{S(init)\} \rangle \models ((\exists \alpha)^*.G)(init) \qquad (5)$$

We now show that the use of procedural rules in the formalization of the dynamic axioms actually weakens the deductive capabilities of the agent formalized. This can be explained by the following simple example. Suppose we have the following dynamic axioms:

$$C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

$$\neg C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

and suppose the goal is D and the initial situation does not specify the truth value of C. It is easy to see that in the $\mathcal{ALC}$ formalization (corresponding to Rosenschein's framework) the answer to the planning problem is yes (the desired plan consists of the action R), while in the $\mathcal{ALCK}$ framework the answer to the planning problem is no.

This is precisely due to the different formalization of the dynamic axioms: In the $\mathcal{ALC}$ setting, the agent is able to conclude that he is able to perform action R, since in the current *state of the world* either C or $\neg C$ holds. Conversely, in the $\mathcal{ALCK}$ framework, in the *epistemic state* of the agent neither C nor $\neg C$ holds, since the agent *does not know* the truth value of C, therefore he concludes that he is not able to perform action R.

Let us now slightly modify the above example. Suppose we have the following dynamic axioms:

$$C \sqsubseteq \exists R_1.\top \sqcap \forall R_1.D$$

$$\neg C \sqsubseteq \exists R_2.\top \sqcap \forall R_2.D$$

and again suppose that the goal is D and the initial situation does not specify the truth value of C. Now, there is no *known* sequence of actions leading to the state satisfying the goal, yet in the $\mathcal{ALC}$ setting the answer to the planning problem is yes, whereas it is still no in the epistemic framework.

This example highlights the fact that in the $\mathcal{ALCK}$ setting *only constructive proofs are taken into consideration*, in the sense that the entailment (3) holds only if there exists a *known* sequence of actions leading to the state satisfying the goal. That is, if (3) holds then we are always able to extract an effective plan.

Conversely, it is easy to see that, if in the epistemic formalization of actions the entailment (3) holds, then the entailment (5) holds in the $\mathcal{ALC}$ setting. Therefore, with respect to the planning problem, the epistemic framework based on $\mathcal{ALCK}$ is a *sound and incomplete* approximation of the non-epistemic one. That is, the use of the epistemic operator in the formalization of actions allows for a *principled* weakening of the deductive capabilities of the agent.

5 THE MOBILE ROBOT "TINO"

Our approach to reasoning about actions has been implemented on the *Erratic* base [Konolige,1995]. The robot, that, as mentioned, has been 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 dates back to the architecture of the robot Shakey, in which the planning system was STRIPS. However, the Erratic base allows for an effective combination of both horizontal and vertical decomposition (see [Brooks,1986]). By a horizontal decomposition the system can react immediately in dynamic environments. While a vertical decomposition is in the relationship between the module which is responsible for planning and the underlying fuzzy controller. Each of these modules has its own representation of the environment and of the plan. The communication between the two modules is realized by a plan execution module. Thus, as in [Gat,1992], we have a heterogeneous architecture, since we have implemented our planning system and then we have connected it with the existing controller.

Below we sketch 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 [Borgida *et al.*,1989], a well-known system based on DLs, to take advantage of an efficient and reliable reasoning system. However, the language for

representing knowledge is less expressive than the DL we have considered so far. Nonetheless, we obtain an interesting setting where the plan can be generated in polynomial time.

More specifically, we use a subset of the language $\mathcal{ALCK}$, corresponding to some of the constructs available in CLASSIC, that we write for ease of notation using $\sqcap$ for `AND`, $\forall$ for `ALL` and $\exists R.\{a\}$ for `FILLS`.

Static axioms are expressed either as inclusion assertions or as concept definitions, written $\doteq$ and interpreted as necessary and sufficient conditions (see for example [Buchheit *et al.*,1994]). In both cases cycles are not allowed. Dynamic axioms are represented as CLASSIC *rules* denoted with $\mapsto$.

Each rule is thus written as

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

and can be read as follows: For each named individual i classified under C, connect i to the individual a through the role R, and classify a under D. Therefore a is an individual denoting the state reached as a result of the execution of action R.

Notice that we are forced to use the `FILLS` (written $\exists R.\{a\}$) construct because we cannot express $\exists \mathbf{K} R.\top$ in CLASSIC. This gives us a sound implementation as long as for each action R the preconditions C in dynamic axioms involving R are disjoint from each other. Indeed, this condition implies that in every state at most one of the preconditions C for each action R is satisfied, consequently the only concept that holds in an R-successor, obtained by applying the dynamic axiom, is D.

With the above described representation, computing the FOE is done in polynomial time, because the number of individuals is at most linear in the number of rules, and the condition for the application of a rule can be checked in polynomial time. Notice that we cannot compute the FOE of a knowledge base Σ in the general case using CLASSIC, because there is no other way to state the existence of a named individual (representing a successor state) other than explicitly naming it through the construct `FILLS`.

The plan is extracted by the explanation facility provided with the rule mechanism of the system, which allows for an automatic generation of a graph (essentially a part of the FOE) with all the paths from the initial state to the states that satisfy the goal. The plan to be sent to the robot is then selected by finding the path (between the initial state and the states that satisfy the goal) which is minimal in terms of the number of transitions (actions).

Example 5.1 Given the map shown in Fig. 5, referring to an office environment constituted by rooms, doors and corridors, we can describe it through the following knowledge base:

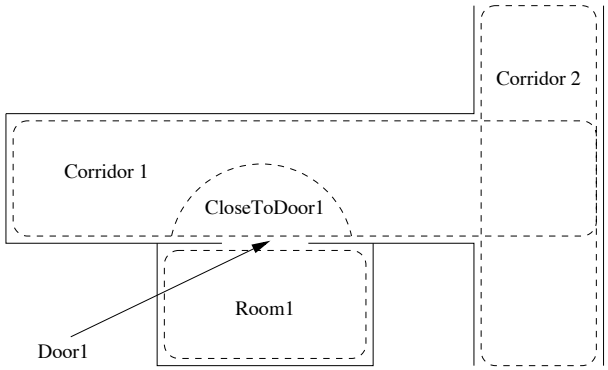

Figure 5: A simple environment

$$
\begin{array}{lcl}
Corridor1 & \sqsubseteq & Corridor \\
Corridor2 & \sqsubseteq & Corridor \\
Room1 & \sqsubseteq & Room \\
CloseToDoor & \doteq & Corridor \sqcap \forall NextDoor.Door \sqcap \\
 & & \exists NextDoor.\top \\
CloseToDoor1 & \doteq & CloseToDoor \sqcap Corridor1 \sqcap \\
 & & \exists NextDoor.\{xDoor1\}
\end{array}
$$

$$
\begin{array}{lcl}
Corridor1 & \mapsto & \exists FollowC1ToD1.\{xCloseToDoor1\} \sqcap \\
 & & \forall FollowC1ToD1.CloseToDoor1 \\
Corridor1 & \mapsto & \exists FollowC1ToC2.\{xCorridor2\} \sqcap \\
 & & \forall FollowC1ToC2.Corridor2 \\
CloseToDoor1 & \mapsto & \exists EnterD1.\{xRoom1\} \sqcap \\
 & & \forall EnterD1.Room1 \\
Room1 & \mapsto & \exists ExitD1.\{xCloseToDoor1\} \sqcap \\
 & & \forall ExitD1.CloseToDoor1
\end{array}
$$

$$Door(xDoor1)$$
$$Corridor1(xCorridor1)$$

Let us now highlight the use of static axioms for the description of knowledge about environments and for the classification of states. In the above example, the concept *CloseToDoor* formalizes the property of being in a corridor and close to a door, while the concept *CloseToDoor1* represents the property of being close to a particular door (*Door1*). *NextDoor*, which is assumed to be a functional role, is used to describe a static property of a portion of a corridor.

The action graph relative to the above knowledge base is given in Fig. 6. Notice that with the rule propagation mechanism we can produce many edges of the graph with a single rule. In fact, suppose there are many doors in *Corridor*1, with the only rule that describes the action $FollowC1ToC2$, we can connect in the graph all the states $xCloseToDoor(i)$ to the state $xCorridor2$ with edges labeled with the action $FollowC1ToC2$. This is due to the fact that we can write static axioms such that $CloseToDoor1 \sqsubseteq Corridor1$.

Moreover, static axioms allow us to write postconditions of actions in a simple way. For example in the de-

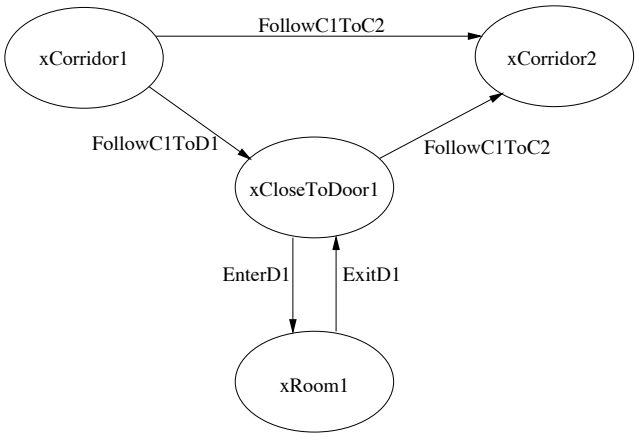

Figure 6: The action graph

scription of the action $ExitD1$ we have only mentioned $CloseToDoor1$ as postcondition, instead of writing all the postconditions (in this case also $Corridor1$) explicitly.

The ability to provide a taxonomic representation of the environment not only provides a more compact way to specify the knowledge about the problem domain, but also 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 the implementation of a module that takes as input a topological representation of the map and generates a CLASSIC knowledge base.

5.2 PLAN EXECUTION

The planning system is activated by a plan execution module which provides the connection to the robot software. The control of the robot is achieved by means of a fuzzy controller [Saffiotti *et al.*,1995] which takes care of obstacle avoidance while the robot is trying to achieve a high-level goal such as reaching the next door in the corridor.

More specifically, the plan execution module turns the sequence of actions, expressed as CLASSIC roles, into a specification for the control software. The control system drives the robot low-level movement commands by means of control schemata, called *behaviors*, which specify both how to implement high-level actions and how to perform several kinds of reactive control, such as obstacle avoidance. Therefore, the simultaneous activation of reactive behaviors, and of the ones implementing actions is usually requested to perform a task. For example, the activation of the behaviors *Avoid Obstacle*, *Keep Off* and *Follow Corr*, is used to realize navigation within a corridor. The actual execution can be viewed as the *blending* of such behaviors, based on the information acquired through the sensors (see [Saffiotti *et al.*,1995] for further details). The idea is related to the notion of plan-as-communication [Agre and Chapman,1990], since the plan is not mechanically executed, but is used to decide which behavior to activate.

As each module has its own representation of information, the exchange of information between the two layers is a critical aspect. The planning system sends a plan to the control system, the latter can detect a plan failure and reply to the former with a justification for the failure (such as "door closed") so that the planning system can re-plan after updating the knowledge base.

Example 5.2 We can describe the fact that the robot can enter in a room only if it is close to an open door by adding the following declarations to the knowledge base in Example 5.1.

$$\begin{aligned} CloseToOpenDoor &\doteq CloseToDoor \sqcap \\ &\quad \forall NextDoor.Open \\ CloseToOpenDoor1 &\doteq CloseToOpenDoor \sqcap \\ &\quad CloseToDoor1 \end{aligned}$$

$$\begin{aligned} CloseToOpenDoor1 \mapsto\ & \exists EnterD1.\{xRoom1\} \sqcap \\ & \forall EnterD1.Room1 \end{aligned}$$

$$Door(xDoor1)$$
$$Open(xDoor1)$$

Now, in presence of a failure during a plan execution, caused by the fact that door $xDoor1$ is closed, it is sufficient to update the knowledge base, by retracting the assertion $Open(xDoor1)$, so that the new rule involving *EnterD1* cannot be applied and the associated edge will not be in the action graph. If there is another door to access the room, it may be possible to select an alternative plan to reach it.

6 CONCLUSIONS

The goal of our work was to devise a principled and practically feasible realization of a KR&R approach to reasoning about actions 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 research on reasoning about actions, the basis for our work has been the formal correspondence between Propositional Dynamic Logics and Description Logics. This has lead 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 use of rules instead of axioms both reduces the computational complexity of the planning problem and (under some restrictions) allows for the implemen-

tation of the theoretical framework for reasoning about actions in an efficient KR system (CLASSIC).

As compared with other approaches to action planning for mobile robots, 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 and accommodate the changes in the environment. To this end, the possibility of structuring the representation of environments using the features of a DL representation language plays a crucial role.

The implementation developed so far is mainly concerned with the position of the robot and its movement capabilities. We are currently working at several extensions of the proposed framework, that will enable Tino to address more complex scenarios. To this end, we can exploit the notion of epistemic state of the agent, using it 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 [Donini *et al.*,1995] that a slight modification in the semantics of the epistemic operator allows for the representation of default rules in $\mathcal{ALCK}$, thus allowing for the formalization of the notion of default persistence of knowledge, realized through the use of defaults. This property, together with the notion of default persistence of ignorance encoded in the semantics, allows for a formalization of the commonsense law of inertia, realized through a "small" number of epistemic axioms (formalizing default rules).
2. **Sensing.** The possibility of representing the epistemic state of the agent allows for a very simple formalization of sensing (or knowledge-producing) actions. Indeed, the semantic of the epistemic operator in $\mathcal{ALCK}$ embodies the principle of minimal learning, or default persistence of ignorance [Scherl and Levesque,1993], thus allowing for a simple treatment of actions whose execution only changes the knowledge of the agent without affecting the state of the world.

Acknowledgments

The work has been supported by Italian MURST. Riccardo Rosati acknowledges the AI Center of SRI International, Menlo Park, for the support provided during his visiting period.

References

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

[Artale and Franconi, 1994] Alessandro Artale and Enrico 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.

[Borgida *et al.*, 1989] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori 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.

[Borgida, 1992] Alex Borgida. Towards the systematic development of description logic reasoners: Clasp reconstructed. *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, 1992.

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

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Buchheit *et al.*, 1994] Martin Buchheit, Francesco M. Donini, Werner Nutt, and Andrea 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.

[Calvanese, 1996] Diego 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.

[De Giacomo and Lenzerini, 1994] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 205–212, 1994.

[De Giacomo and Lenzerini, 1995a] Giuseppe De Giacomo and Maurizio Lenzerini. Enhanced propositional dynamic logic for reasoning about concurrent actions (extended abstract). In *Working notes of the AAAI 1995 Spring Symposium on Extending Theories of Action: Formal and Practical Applications*, pages 62–67, 1995.

[De Giacomo and Lenzerini, 1995b] Giuseppe De Giacomo and Maurizio 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.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea 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.

[Donini *et al.*, 1994] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Queries, rules and definitions. In *Foundations of Knowledge Representation and Reasoning*. Springer-Verlag, 1994.

[Donini *et al.*, 1995] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Non first-order features in concept languages. In *Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA-95)*, Lecture Notes In Artificial Intelligence. Springer-Verlag, 1995.

[Gat, 1992] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. *Proceedings of the 11th National Conference on Artificial Intelligence*, 1992.

[Kaelbling and Rosenschein, 1995] L. Kaelbling and S. Rosenschein. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.

[Koehler, 1994] Jana 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.

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

[Kozen and Tiuryn, 1990] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

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

[McAllester, 1991] David McAllester, 1991. Unpublished Manuscript.

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

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

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

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

[Scherl and Levesque, 1993] Richard Scherl and Hector 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.

[Schild, 1991] Klaus 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, 1991.

[Simmons, 1992] Reid Simmons. An architecture for coordinating planning, sensing and action. *Proceedings of the 11th National Conference on Artificial Intelligence*, 1992.