CLASSIC planning for mobile robots

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
{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. Our proposal is considered "classic" since it is in the style of deductive planning and it relies upon the reasoning capabilities of description logics, that are realized in the system CLASSIC. From an epistemological point of view 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-based 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

In Artificial Intelligence there has always been a great interest in the design of agents that can exhibit different forms of intelligent behavior. A mobile robot can be regarded as an intelligent agent, that is designed both to achieve high-level goals and to be able to promptly react and adjust its behavior based on the information acquired through the sensors. In the recent past, much attention has been directed to the reactive capabilities of agents in general, and mobile robots in particular. Reactive capabilities are often necessary to cope with the uncertainties of the real-world. However, the capabilities of reasoning about actions and high-level planning are important as well. In this respect, we consider two aspects particularly relevant: the integration of the reactive and planned activities, and the possibility of using a general knowledge representation system, where one can design the agent by relying on the general purpose tools provided by the system. In this paper we mostly focus on the latter issue. Indeed we address high-level planning for a mobile robot within the framework of Propositional Dynamic Logics (PDLs), and we rely on a tight correspondence that exists between PDLs and Description Logics (DLs). In this way we are able to relate planning in PDLs with an implementation that uses the knowledge representation system CLASSIC [1, 2], which is based on DLs. However, we also address the integration of the proposed framework within an actual mobile robot. Indeed, starting from a logical formalization we arrive at the realization of an *Erratic*-based mobile robot, which is equipped with wheels and sonars [10]. We named our mobile robot agent "Tino" and demonstrated it at the 1995 Description Logic Workshop.

The natural way to approach the design of a system for plan generation in a knowledge representation system has its roots in the work on deductive planning. The idea is that a plan can be generated by finding an existence proof for the state where the desired goal is satisfied [9]. However, this approach has mostly been considered at a theoretical level, since the computational cost of deriving a plan from a logical specification has always been considered too high. We face this difficulty, that arises in the PDLs-framework as well, by making use of the mentioned correspondence to take advantage of the work on DLs, which has paid a special attention to the trade-off between expressivity and efficiency of reasoning.

The basic elements of our work originate from the proposals in [13, 5] of using the PDLs framework for reasoning about actions and deductive planning. 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. In our formalization we closely follow this setting. However, we rephrase the above setting in DLs to enable the robot to reason efficiently. Notably, from the body of research on DLs it is known 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 [7] we have been able to define a setting which provides both a novel epistemic representation of dynamic axioms and a weak form of reasoning, which makes the implementation of the deductive planning approach computationally feasible.

The paper is organized as follows. In Section 2, we present the general framework for the representation of dynamic systems. In Section 3, we address our specific way to represent and reason about actions, pointing out the connections with a description logic based knowledge representation system. Finally, in Section 4, we describe the mobile robot "Tino", which includes the implementation in CLASSIC of the planning component, a simple monitor 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 basis of our framework for representing dynamic systems. Such a framework is essentially that of dynamic logics [5, 13], but we make use of the correspondence between propositional dynamic logics (PDLs) and description logics (DLs). The correspondence, first pointed out by Schild [15], 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 [15, 4]. For our purposes it suffices to consider DLs concepts and roles as syntactic variants of PDLs formulae and actions respectively. We present our proposal using the notation of DLs, in order to make it easier to relate the formalization to the actual implementation in CLASSIC, which is a DLs based knowledge representation system.

Following a common approach in reasoning about actions, dynamic systems are modeled

in terms of state evolutions caused by actions. A *state* is a *complete* description (wrt some logic/language) of a situation the system can be in. *Actions* 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* represents a state, and is labeled by the properties that characterize the state.
- Each *arc* represents a state transition, and is labeled by 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 such knowledge is phrased in *axioms* of some logic (e.g. Dynamic Logic [13] or Situation Calculus [12]). These axioms select a subset of all possible transition graphs, which are similar since they all satisfy the same axioms, but yet different wrt those properties not imposed by the axioms. The actual behavior of the system is indeed denoted by one of the selected graphs, however which one is not known. 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 [13] two kinds of axioms are distinguished:

- *Static axioms* (also called "domain constraints") 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* are introduced to represent the changes actions bring about. Intuitively they have the following form:

  $$C \stackrel{R}{\Longrightarrow} D$$

  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.

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?". Assuming actions to be deterministic this is captured by the following logical implication (here we phrase it in PDLs/DLs):

$$\Gamma \cup \{S(init)\} \models (\exists \alpha^*.G)(init) \tag{1}$$

where: (i) Γ is the set of both static and dynamic axioms representing the (partial) knowledge on the system; (ii) S is a formula representing the (partial) knowledge on the initial situation (state) which is denoted by $init$; (iii) G is a formula representing the goal, which is, in fact, a (partial) description of the final state one wants to reach; (iv) $\exists \alpha^*.G$ is concept (i.e. a formula in PDLs), which holds in $init$, and expresses the fact that there exists a finite sequence of actions (here α stands for any action) 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 on the system (say a few laws/axioms we know the system obeys) and yet be able to make several non-trivial inferences. Unfortunately, this generality is paid by a high computational cost (typically unrestricted PDLs/DLs are EXPTIME-complete [11, 4]).

In order to lower the computational cost of reasoning, we are going to admit in our formalism only restricted forms of axioms. In particular, dynamic axioms will make a special use of *procedural rules*, whose semantics can be formalized through the *epistemic operator* introduced in [7, 8] and interpreted in terms of minimal knowledge.

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. Such a graph, called in the following *partial transition graph*, gives us an incomplete description of the actual transition graph where certain states and transitions may be missing, and the properties of the states in the graph may be only partially specified.

3 Reasoning about actions: our proposal

In our ontology, properties of states are represented as concept expressions of DLs. This means that a concept expression denotes a property that may hold in a state.

Actions are represented as DLs roles. In fact we distinguish two kinds of roles: *static-roles* which represent the usual notion of role in DLs and can be useful to specify further properties of states; *action-roles* which denote actions (or better state transitions caused by actions) and are used in a special way.

The initial situation is denoted by $S(init)$ where $init$ names the initial state and S is a concept describing our knowledge on the initial state.

The behavior of the agent is described by means of both static axioms and dynamic axioms. In principle, axioms could be represented as inclusion statements of the most general form:

$$C \sqsubseteq D$$

where C and D are concepts of DLs. However, reasoning with general inclusions is intractable and restrictions are normally considered. For example, concepts on the left-hand side are typically required to be primitive, and cycles (recursive inclusions), which are especially problematic from the computational point of view, are not allowed (see for example [3, 4]).

Therefore we model static axioms as acyclic inclusion assertions and concept definitions, not involving action-roles, whereas we model the dynamic axioms, which are inherently cyclic, by making a special use of procedural rules. This has the consequence of weakening the reasoning capabilities and thus gaining efficiency. By exploiting the epistemic interpretation of procedural rules [7, 8] we formalize dynamic axioms through epistemic sentences of the form:

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

which can be intuitively interpreted as: "if C holds in the current state x of the partial transition graph, then there exists an R-successor y in the partial transition graph, and for all the R-successors of x (and hence for y) D holds".

Given a knowledge base including both static and dynamic axioms, as above described, and a specification of the initial state, the partial transition graph can be generated as follows. We start from the initial state $init$, which is going to be the first state included in the partial transition graph. On this state certain properties, namely those denoted by S, are known

to hold. Hence we can apply the dynamic axioms whose preconditions C hold in $init$ – i.e. C must be implied by S plus the static axioms. By applying a dynamic axiom in $init$ we get an R-successor s in the partial transition graph, and in the new state s we know that D holds. Again we can apply the dynamic axioms whose preconditions hold in s, and so on. Note that if two states of the partial transition graph have exactly the same properties, i.e. the same concepts hold, then we can collapse the two states into one. Therefore, to avoid redundancies, we can assume that in the final partial transition graph all states are in fact distinguished. Observe that state of the partial transition graph are (partial) descriptions of states of the actual transition graph, hence we are requiring that no two such descriptions are logically equivalent. We shall see that the notion of first-order extension exactly captures this process. Moreover it ensures us that there exists a unique partial transition graph.

Notably the dynamic axioms cannot be used in the reverse direction for contrapositive reasoning, and this weakening lowers the computational cost of reasoning in our formalism.

We remark that, although at first sight it may not be apparent, in our formalization actions are deterministic. Indeed the only existential we allow for actions is unqualified ($\exists \mathbf{K}R.\top$). Hence we do not have the ability to say that something holds in one R-successor state and doesn't in another one.

We can now formulate the planning problem in our setting. In the following, we use Γ_S to indicate the set of static axioms and Γ_D to indicate the set of dynamic axioms; S stands for the concept describing the properties of the initial state and G stands for the concept corresponding to the goal.

A plan exists for the goal G if there exists a finite sequence of actions which, from the initial state, leads to a state satisfying the goal. This condition is expressed by the logical implication (1), which can be rephrased in our setting as follows:

$$\langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle \models ((\exists \alpha)^*.\mathbf{K}G)(init) \tag{2}$$

where $(\exists \alpha)^*\mathbf{K}G$ 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. Condition (2) checks for the existence of a state of the partial transition graph in which the goal is satisfied. It can be shown that condition (2) holds iff for each epistemic model $\mathcal{M}$ for Σ, there exists an individual x such that $\mathcal{M} \models G(x)$.

Formally, the semantics of a knowledge base containing epistemic sentences is given in terms of *preferred models*, in which the objective knowledge is minimized (we refer to [7, 8] for a detailed presentation of these notions). Generally, the knowledge base $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(init)\}\rangle$ has many preferred models, which are distinguishable even up to renaming of individuals. However to characterize reasoning, we can follow the approach defined in [7], and resort to a notion of *first-order extension* (FOE) of a 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 first-order assertions which are consequences (up to renaming of individuals) of the epistemic sentences describing the dynamic axioms, thus providing a characterization of the knowledge that is shared by all the preferred models of Σ. $FOE(\Sigma)$ can be computed by an algorithm which, starting from the initial individual $init$, applies to each named individual the dynamic axioms in the set Γ_D which are triggered by such individual. A new individual is thus generated, unless an individual with the same properties had already been created. In this way the effect of the dynamic axioms is computed, thus obtaining a sort of "completion" of the knowledge base, which captures the intuition underlying partial transition graphs.

Since the first-order extension of Σ represents the information which must hold in *any* preferred model for Σ, up to individual names, we can solve the planning problem (2) by solving the following existential entailment problem on the first-order knowledge base $FOE(\Sigma)$:

$$FOE(\Sigma) \models \exists x.G(x) \tag{3}$$

In general the number of dynamic axiom applications required to verify the above condition is exponential in the number of dynamic axioms appearing in Σ, since the number of new individuals generated in the first-order extension is 2^n in the worst case, where n is the number of dynamic axioms in Σ. However, under the assumption that for each action-role R the preconditions C in the dynamic axioms involving R are disjoint from each other, the number of individuals generated by the algorithm which computes $FOE(\Sigma)$ is at most equal to the number of dynamic axioms, since every application of the same dynamic axiom generates the same individual.

4 The mobile robot "Tino"

Our approach to reasoning about actions has been practically tested on the *Erratic* base [10] and demonstrated at the 1995 Descriptions Logic Workshop. The basic approach to the software architecture of the robot allows one to reach a good balance between reactive behavior and high-level goal achievement. Below we sketch the basic ideas of the implementation of the planning component and of its connection to the underlying software layer, which implements the robot's reactive behavior.

In the implementation of planning, static axioms are modeled as CLASSIC inclusion assertions and concept definitions (cycles are not allowed), while dynamic axioms are modeled as CLASSIC rules of the form

```
C => (AND (ALL R D) (FILLS R a))
```

where a is an individual denoting the state reached as a result of the action execution. In this way we identify all successor states resulting by applying a dynamic axiom, which results in a sound reasoning method as long as the following condition holds: *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. Therefore the only concept that holds in a R-successor, obtained by applying the dynamic axiom $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \sqcap \forall R.D$, is D. Hence, we can identify all such successor states and give an explicit name (using `FILLS`) to such a successor without compromising the soundness of reasoning.

As mentioned above, this condition causes the generation of a number of new known individuals which is at most linear in the number of rules, thus keeping the CLASSIC implementation tractable. On the other hand, this condition must hold for the correctness of our implementation. In fact, in the CLASSIC setting, rules are actually used to extend the knowledge base just like in the first-order extension. Since in the CLASSIC implementation we use the `FILLS` construct to simulate the existential quantification on epistemic roles, each individual name must appear in the rules, therefore the cardinality of the set of individuals in the CLASSIC knowledge base is linearly bounded to the number of rules. Consequently, we cannot compute the FOE of a knowledge base Σ in the general case using CLASSIC.

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 the partial transition

graph) with all the paths from the initial state to the final state. The plan to be sent to the robot is then selected by finding the path between the initial state and the final state which is minimal in terms of the number of actions.

The integration of the planning capabilities with the reactive behavior of the robot is realized by a fuzzy controller 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. A critical aspect of this kind of multi-level approach to the robot software architecture is in the exchange of information between the different layers. In our case we need to represent the information about the map into a knowledge base of static axioms and to represent the action descriptions into a set of rules, which requires to turn the low-level representation of the robot into a CLASSIC knowledge base. To this end we have implemented a module that takes as input the internal map and generates the knowledge base.

Moreover, we have implemented a simple monitor which calls the planner, activates the underlying execution levels and controls the execution of the plan. This is achieved by combining each action with the reactive behaviors for obstacle avoidance, through the underlying "blending" mechanism (see [14]). By setting a time out for the execution of each action the monitor detects the plan failure and in this case provides a justification for the failure such as for example "door closed". In such a case the system can re-plan by updating the knowledge base and, consequently, the internal representation.

5 Conclusion

The goal of our work was to exploit the reasoning services offered by a knowledge representation system based on description logics to the task of plan generation.

We have presented a formalization of planning based on Description Logics and described an implementation of the proposed framework in the system CLASSIC. Previous work [6] aimed at using CLASSIC for reasoning about plans, by introducing a formalism for describing plans, while the present work is focussed on plan generation and relies on the standard features of the system. Our implementation is actually used to plan the actions of the mobile robot *Erratic*, capable of integrating reacting behavior with action execution. We remark that the current implementation does not need to make use of a persistence assumption when computing a successor state, since all the relevant properties are either static or explicitly specified. However, in order to deal with more complex scenarios we are further developing our approach to enforce this kind of assumptions.

One original contribution of our work is in the idea of using the rule mechanism, and its associated interpretation in terms of minimal knowledge, to weaken the assumptions underlying a first-order formalization of actions. In this way a plan can be obtained by a forward reasoning process, that is weaker than ordinary deduction, but semantically justified and computationally feasible.

References

[1] 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.

[2] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[3] 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.

[4] G De Giacomo and M. 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.

[5] G. De Giacomo and M. 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.

[6] P. Devambu and D. Litman. Plan-based terminological reasoning. In James Allen, Richard Fikes, and Erik 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.

[7] 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.

[8] F. M. Donini, D. Nardi, and R. 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.

[9] C. C. Green. Theorem proving by resolution as basis for question-answering systems. In *Machine Intelligence*, volume 4, pages 183–205. American Elsevier, 1969.

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

[11] 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.

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

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

[14] A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. Technical report, SRI International, Menlo Park, CA, 1995.

[15] 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, 1991.