Minimal Knowledge Approach to Reasoning about Actions and Sensing

Giuseppe De Giacomo and **Riccardo Rosati**
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{degiacomo,rosati}@dis.uniroma1.it

Abstract

We present an autoepistemic approach for reasoning about actions in the presence of incomplete information and sensing. Specifically, we introduce a logical formalism that combines a very expressive logic of programs, the modal mu-calculus, with a minimal knowledge modality. We show that reasoning in such a formalism can be done by integrating model checking for modal mu-calculus and propositional inference. This allows for exploiting existing model checking techniques and systems for sophisticated forms of reasoning about actions, without renouncing to deal with incomplete information about the dynamic system.

1 Introduction

Research in Cognitive Robotics [22; 23] is forcing the area of reasoning about actions to go through a "reality check". It has shown that, when one wants to equip an actual robot with the ability of reasoning about its actions, it is essential to take into account that the robot normally operates in an environment which it only partially knows, and that it must dynamically acquire new information through its sensors when needed [17]. Moreover, the basic reasoning task of projection is in general not sufficient to reason about sophisticated aspects of the robot's behavior, such as being always responsive to requests of other agents, or guaranteeing the successful termination of certain activities. Finally, reasoning about actions must be effective, i.e., reasonably efficient, in this generalized setting.

In this paper we propose a logical formalism for reasoning about actions which has the potentiality of meeting all the requirements above. Specifically, we define a variant of modal mu-calculus [24], a logic of programs that subsumes both propositional dynamic logics, such as standard PDL and ΔPDL [13], and branching time temporal logics such as CTL and CTL* [9]. Modal mu-calculus is used in the verification of concurrent systems [12; 20], and for this task several automated model checking techniques and systems have been developed [24; 3; 19].

We extend modal mu-calculus with an autoepistemic modal operator in order to represent and reason about the epistemic state of the robot. Autoepistemic operators have already been introduced for reasoning about actions e.g. in [14; 18]. Here, following [6; 7], we use a minimal knowledge modality in a way that strongly characterizes how the deliberative behavior of the robot is modeled: the robot may perform an action if it *knows* that the preconditions for that action hold, not simply if the preconditions are true. Similarly, the effects of an action of interest for the robot are only the ones the robot is aware of, i.e. the effects that change its epistemic state. This is obtained by specifying what a robot knows after an action, instead of specifying what is true after an action. In this approach, the robot follows the changes in the world through the changes in its epistemic state only. The minimal knowledge modality is also used to provide a natural formalization of *sensing actions*, i.e., actions that allow the robot to know whether a certain property holds in the current state of the world [17; 11; 2].

The special use of the minimal knowledge modality, that we require in the axioms specifying preconditions and effects of actions, forces a strong uniformity on the models of the dynamic system specification. This uniformity allows for representing all models by means of a single transition graph, whose nodes correspond to epistemic states of the robot, and transitions reflect how the robot's epistemic state changes by executing actions.

The proposed extension inherits from modal mu-calculus the ability of expressing very general dynamic and temporal properties. Moreover, by exploiting the possibility of representing all models of a dynamic system specification as a single graph, it becomes possible to adapt model checking techniques for the modal mu-calculus to our setting. Essentially, such model checking techniques are used to visit the graph in a suitable fashion, checking validity (instead of truth) of propositional formulae on single states, while traversing the graph.

2 Logical formalism

The technical background of our proposal is constituted by a logical formalism $\mathcal{L}$ that originates from a suitable integration of modal mu-calculus and autoepistemic description logics (see respectively [24; 9] and [8] for an introduction to these formalisms). The basic elements of $\mathcal{L}$ are a finite set of actions Act, a countable set of propositions $Prop$, and a countable set of propositional variables Var.

Formulae of the formalism are divided in two layers:

- *state description formulae*:
$$p ::= A \mid p_1 \wedge p_2 \mid \neg p$$
- *dynamic formulae*:
$$\phi ::= \mathbf{k}p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid [a]\phi \mid X \mid \mu X.\phi$$
where $A \in Prop$, p is a state description formula, $a \in Act$ and $X \in Var$. The formula ϕ in $\mu X.\phi$ must be syntactically monotone in X, that is the variable X must be in the scope of an even number of negations.

We use the usual abbreviations $\vee, \supset$, tt, ff (the last two denoting tautology and contradiction respectively), and also the abbreviations $\langle a\rangle\phi \doteq \neg[a]\neg\phi$ and $\nu X.\phi \doteq \neg\mu X.\neg\phi[X/\neg X]$ where $[X/\neg X]$ denotes the syntactic substitution of X by $\neg X$.

We give the semantics of $\mathcal{L}$ by first fixing once and for all a countable-infinite set $\mathcal{S}$ of *state names* which will constitute the interpretation domain of $\mathcal{L}$. We assume to have a set of constants $Const \subseteq \mathcal{S}$ that are used to denote state names.

A *pre-interpretation* $\mathcal{I}$ is a function over $\mathcal{S}$ which assigns to each constant in $Const$ the corresponding state name, i.e. $s^{\mathcal{I}} = s$; to each atomic proposition in $Prop$ a subset of $\mathcal{S}$, i.e. $A^{\mathcal{I}} \subseteq \mathcal{S}$; and to each action $a \in Act$ a functional relation over $\mathcal{S}$, i.e. $a^{\mathcal{I}} \subseteq \mathcal{S} \times \mathcal{S}$, with the restriction that for every $s, s', s'' \in \mathcal{S}$ if $(s,s') \in a^{\mathcal{I}}$ and $(s,s'') \in a^{\mathcal{I}}$ then $s' = s''$. In addition, the union of the relations interpreting the actions is backward functional, i.e. for every $s, s', s'' \in \mathcal{S}$ if $(s',s) \in \cup_{a\in Act} a^{\mathcal{I}}$ and $(s'',s) \in \cup_{a\in Act} a^{\mathcal{I}}$ then $s' = s''$. Pre-interpretations are extended to state description formulae as follows:

$$\begin{array}{rcl} (p_1 \wedge p_2)^{\mathcal{I}} & = & p_1^{\mathcal{I}} \cap p_2^{\mathcal{I}} \\ (\neg p)^{\mathcal{I}} & = & \mathcal{S} - p^{\mathcal{I}} \end{array}$$

A *valuation* ρ is a function from Var to a subset of $\mathcal{S}$ such that $\rho(X) \subseteq \mathcal{S}$ for every $X \in Var$. Given a valuation ρ and $\mathcal{E} \subseteq \mathcal{S}$,, we denote by $\rho[X/\mathcal{E}]$ the valuation obtained from ρ by changing to $\mathcal{E}$ the subset assigned to the variable X.

A *interpretation* $\mathcal{W}$ is a set of pre-interpretations over $\mathcal{S}$. We define interpretations of state formulae and actions respectively as:

$$\begin{array}{rcl} p^{\mathcal{W}} & = & \cap_{\mathcal{I}\in\mathcal{W}}\, p^{\mathcal{I}} \\ a^{\mathcal{W}} & = & \cap_{\mathcal{I}\in\mathcal{W}}\, a^{\mathcal{I}} \end{array}$$

Interpretations and valuations are used to interpret dynamic formulae as follows:

$$\begin{array}{lcl} (\mathbf{k}p)^{\mathcal{W}}_{\rho} & = & p^{\mathcal{W}} \\ (\phi_1 \wedge \phi_2)^{\mathcal{W}}_{\rho} & = & (\phi_1)^{\mathcal{W}}_{\rho} \cap (\phi_2)^{\mathcal{W}}_{\rho} \\ (\neg\phi)^{\mathcal{W}}_{\rho} & = & \mathcal{S} - \phi^{\mathcal{W}}_{\rho} \\ ([a]\phi)^{\mathcal{W}}_{\rho} & = & \{s \in \mathcal{S} \mid \forall s'.(s,s') \in a^{\mathcal{W}} \supset s' \in \phi^{\mathcal{W}}_{\rho}\} \\ X^{\mathcal{W}}_{\rho} & = & \rho(X) \\ (\mu X.\phi)^{\mathcal{W}}_{\rho} & = & \cap\{\mathcal{E} \subseteq \mathcal{S} \mid \phi^{\mathcal{W}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\} \end{array}$$

In particular we will be interested in closed formulae (formulae with no free variables). Such formulae are interpreted independently from the valuation, hence we will interpret them using an interpretation $\mathcal{W}$ alone: $\phi^{\mathcal{W}}$.

A *knowledge base* Σ is defined as a pair $\Sigma = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a finite set of state description formulae and (closed) dynamic formulae, and $\mathcal{A}$ is a finite set of assertions of the form $\psi(s)$ with ψ either a state description formula or a dynamic formula, and $s \in Const$.

An interpretation $\mathcal{W}$ *satisfies* a formula $\psi \in \mathcal{T}$ iff $\psi^{\mathcal{W}} = \mathcal{S}$. $\mathcal{W}$ satisfies an assertion $p(s) \in \mathcal{A}$ iff $s \in p^{\mathcal{W}}$. $\mathcal{W}$ satisfies a knowledge base $\Sigma = (\mathcal{T}, \mathcal{A})$ iff $\mathcal{W}$ satisfies every formula from $\mathcal{T}$ and every assertion from $\mathcal{A}$.

An interpretation $\mathcal{W}$ is a *model* for Σ iff $\mathcal{W}$ is a maximal set of interpretations satisfying Σ, i.e., for each interpretation $\mathcal{W}'$, if $\mathcal{W} \subset \mathcal{W}'$ then $\mathcal{W}'$ does not satisfy Σ. This corresponds to impose a "minimal knowledge" semantics on the epistemic states of the agent [8]. In fact, each interpretation can be viewed as a Kripke structure in which each pre-interpretation is a possible world, and each world is connected to all worlds in the structure: only structures satisfying Σ and having a maximal set of possible worlds are considered, which maximizes ignorance of the agent in its epistemic states.

Finally, Σ *logically implies* a formula or an assertion σ, written $\Sigma \models \sigma$, iff every model for Σ satisfies σ.

3 Dynamic system representation

In this section we present our framework for representing dynamic systems in the logic $\mathcal{L}$. The framework essentially follows the one presented in [6; 7].

The formalization of a dynamic system is constituted by the following elements.

- *Initial state description* is formed by a finite set of assertions of the form
$$p(s_{init})$$
where p is a state description formula and s_{init} is a constant in $Const$. In fact we may assume that $Const = \{s_{init}\}$.

- *Static axioms* (also known as *state constraints*) are a finite set of state description formulae p, which are assumed valid, defining invariance properties of states.

- *Precondition axioms* specify under which conditions an action can be executed. In our case such a condition depends on the epistemic state of the agent and not on what is true in the world. Precondition axioms are dynamic formulae of the the form:
$$\mathbf{k}p \supset \langle a\rangle\mathbf{k}\,tt$$

- *Effect axioms* specify the effects of an ordinary (i.e., non-sensing) action when executed under certain conditions. Again, in our approach both effects and conditions concern the epistemic state of the agent. Effect axioms are dynamic formulae of the form:
$$\mathbf{k}p_1 \supset [a]\mathbf{k}p_2$$

No special treatment of the frame problem is considered here; we simply make use of frame axioms constituted by effect axioms of the form:

$$\mathbf{k}p \supset [a]\mathbf{k}p$$

- *Sensing effect axioms* are effect axioms of a special form, which specify the outcome of a sensing action. Suppose a_f is a generic sensing action whose effect is to let the agent know the truth value of the property f, where f is any state formula. Also, suppose p is the precondition for the execution of a_f. Such a sensing action is represented in our framework by an usual action precondition axiom $\mathbf{k}p \supset \langle a_f \rangle \mathbf{k}tt$, plus the sensing effect axiom

  $$\mathbf{k}p \supset [a_f](\mathbf{k}f \vee \mathbf{k}\neg f)$$

  which formalizes the fact that, after the execution of a_f, the robot knows whether f holds or $\neg f$ holds.

Finally, for each sensing action a_f, we enforce a *frame axiom schema* of the form:

$$\mathbf{k}\varphi \supset [a_f]\mathbf{k}\varphi$$

which formalizes the fact that all the properties known by the robot before the execution of the sensing action are still known after executing it. Observe that, as a consequence of the frame axiom schema, if the robot already knows the truth-value of f then the sensing action a_f does not have any effect, in the sense that, if the robot knows f $(\neg f)$, then after executing a_f the robot will still know f $(\neg f)$. It is possible to show that the above axiom schema can be represented, without loss of generality, through a finite (linear) number of instances, by replacing φ in the schema with the initial state description and with each effect appearing in effect axioms.

Let Σ be the knowledge base describing the dynamic system as above. We are interested in verifying if the system satisfies a certain dynamic property. Formally, we are interested in logical inference of the form

$$\Sigma \models \phi(s_{init}) \tag{1}$$

where ϕ can be any dynamic formula. As we shall see later, in this way we can deal for instance with the *projection problem*, "given a sequence of actions, does a given state description formula hold in the resulting state?"; the *planning problem*, "is there a sequence of actions such that the goal (a state description formula) holds in the resulting state?"; but also very sophisticated dynamic properties such as *liveness*, *safeness*, etc. that are easily expressed using fixpoint formulae.

4 Reasoning technique

Let us now turn our attention to the problem of computing the logical implication (1).

First of all, an $\mathcal{L}$ knowledge base Σ corresponding to a dynamic system specification has in general many models. When no sensing action is formalized, all models of Σ are isomorphic up to renaming of states, and it is possible to reason about a single model, since it can be shown that all the properties that are expressible in the right-hand side of (1) are independent of such state names.

```
ALGORITHM TG
INPUT: $\Sigma = (\Gamma_S \cup \Gamma_P \cup \Gamma_E, \{p(s_{init})\})$
OUTPUT: $TG(\Sigma)$
PROCEDURE CREATE_NEW_STATE$(s, a)$
begin
   $s'$ = NEW state name;
   $Prop(s') = \{q | (\mathbf{k}p' \supset [a]\mathbf{k}q \in \Gamma_E) \wedge (\Gamma_S \cup L_S(s) \models p')\}$;
   if there exists $s'' \in S$ such that
      $(\Gamma_S \cup L_S(s'))$ and $(\Gamma_S \cup L_S(s''))$ are logically equiv.
   then $L_A(s, a) = s''$
   else begin
      $S_{active} = S_{active} \cup \{s'\}$;
      $S = S \cup \{s'\}$;
      $L_S(s') = Prop(s')$
   end
end
begin
   $S_{active} = \{s_{init}\}$;
   $S = \{s_{init}\}$;
   $L_S(s_{init}) = \{p(s_{init})\}$;
   repeat
      $s$ = choose$(S_{active})$;
      for each ordinary action $a$ do
          if $(\mathbf{k}p \supset \langle a \rangle \mathbf{k}tt) \in \Gamma_P$ and $(\Gamma_S \cup L_S(s) \models p)$
          then CREATE_NEW_STATE$(s, a)$;
      for each sensing action $a_f$ do
          if $(\mathbf{k}p \supset \langle a_f \rangle \mathbf{k}tt) \in \Gamma_P$ and
             $(\Gamma_S \cup L_S(s) \models p)$ and
             $(\Gamma_S \cup L_S(s) \not\models f)$ and
             $(\Gamma_S \cup L_S(s) \not\models \neg f)$
          then begin
                 CREATE_NEW_STATE$(s, a_f^+)$;
                 CREATE_NEW_STATE$(s, a_f^-)$
          end;
      $S_{active} = S_{active} - \{s\}$
   until $S_{active} = \emptyset$;
   return $(S, L_S, L_A)$
end;
```

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

On the other hand, the presence of sensing effect axioms in Σ causes in general the existence of models which structurally differ from each other. This can be intuitively explained by the fact that when the robot uses its sensing capabilities to know whether a certain boolean property p holds, its epistemic state changes according to one of the two possible outcomes of the sensing action. Hence, two different models for Σ represent the two different possible epistemic states resulting from the execution of the sensing action.

We represent all the models of Σ by means of the *transition graph* (TG) of Σ. Roughly speaking, the transition graph is a graph in which:

- each node corresponds to a state and is labeled with a propositional formula representing the properties which are known in such a state;

- each edge is labeled with an action name, and denotes the transition caused by the execution of the corresponding action.

Observe that what the robot knows in the initial state is the set of propositional formulae which are valid in s_{init}, i.e. the set of propositional formulae which are logically implied by $p(s_{init})$. Moreover, what the robot knows after executing an ordinary action is the set of propositional formulae which are logically implied by the postconditions representing the effects of the action execution, while what the robot knows after executing a sensing action is the truth value of the sensed fluent, plus the knowledge of the robot before executing the sensing action.

In this way it can be shown that it is possible in each state to verify whether an action can be executed (that is, whether the preconditions are known by the robot) by simply checking for the validity of the action precondition. This correspondence between the notions of robot's *knowledge* (about propositional properties) and propositional *validity* is exploited in the construction of the transition graph.

Formally, $TG(\Sigma) = (S, L_S, L_A)$, where $S \subseteq \mathcal{S}$ is the set of states which includes s_{init}, L_S is a function assigning a finite set of propositional formulae to each state in S, and L_A is a partial function assigning a state to a pair formed by a state and an action.

Let $\Sigma = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is the set of static axioms (Γ_S), precondition axioms (Γ_P), and effect axioms (Γ_E), and $\mathcal{A} = \{p(s_{init})\}$ is the initial state description, be the dynamic system specification. The transition graph $TG(\Sigma)$ is computed by the algorithm shown in Fig. 1.

In order to compute the transition graph, we replace each sensing action a_f by two special actions a_f^+ and a_f^-. We denote by $\Gamma_E^{\pm}$ the set of effect axioms Γ_E in which those for the sensing action a_f are replaced by:

$$\mathbf{k}tt \supset [a_f^+]\mathbf{k}f \qquad\qquad \mathbf{k}tt \supset [a_f^-]\mathbf{k}\neg f.$$

We also use only a finite number of instances of the frame axiom schemas. We denote by $\Gamma_{IFR}^{\pm}$ the set of axioms

$$\mathbf{k}p \supset [a_f^+]\mathbf{k}\varphi \qquad\qquad \mathbf{k}p \supset [a_f^-]\mathbf{k}\varphi$$

obtained by: (1) instantiating the frame axiom schemas in Γ_{FR} for each propositional formula φ such that either $\varphi(init) \in \Gamma_I$, or $\mathbf{k}\varphi$ is in the postcondition of some effect axiom in Γ_E (i.e., φ such that $\mathbf{k}p \supset [a]\mathbf{k}\varphi$, or $\varphi, \neg\varphi$ such that $\mathbf{k}\top \supset [a_\varphi]\mathbf{k}\varphi \sqcup \mathbf{k}\neg\varphi$ in Γ_E); (2) replacing each sensing action a_f by the two special actions a_f^+ and a_f^-. Finally, we add the axioms in the set $\Gamma_{IFR}^{\pm}$ to Γ_E.

Informally, the algorithm, starting from the initial state s_{init}, iteratively proceeds as follows. First, it finds an action a which can be executed in the current state, by identifying in the set Γ_P a precondition axiom for a whose left-hand side is logically implied by the current knowledge base. Then, it propagates the effects of the action a, which again is based on checking whether the left-hand side of each effect axiom for a 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 a in the current state is computed. A new state (or two new states, if a is a sensing action) 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.

The transition graph is unique, that is, every order of extraction of the states from the set S_{active} produces the same set of assertions, up to the renaming of states. Moreover, the algorithm terminates, that is, the condition $S_{active} = \emptyset$ is eventually reached, since the number of states generated is bounded to the number of different subsets of the set $\mathcal{E} = \{q | \mathbf{k}p' \supset [a]\mathbf{k}q \in \Gamma_E\}$, i.e. 2^n, where n is the number of axioms in Γ_E. Finally, the condition

$$(\Gamma_S \cup L_S(s')) \text{ and } (\Gamma_S \cup L_S(s'')) \text{ are logically equivalent}$$

can be verified by a propositional validity check, as well as the propositional logical implication

$$\Gamma_s \cup L_s(s) \models p$$

Next let us define the *extension of a dynamic formula in* $TG(\Sigma)$ wrt a valuation ρ as follows:

$$\begin{array}{lcl}
(\mathbf{k}p)_\rho^{TG(\Sigma)} & = & \{s \in S \mid \Gamma_s \cup L_s(s) \models p\} \\
(\phi_1 \wedge \phi_2)_\rho^{TG(\Sigma)} & = & (\phi_1)_\rho^{TG(\Sigma)} \cap (\phi_2)_\rho^{TG(\Sigma)} \\
(\neg\phi)_\rho^{TG(\Sigma)} & = & S - (\phi)_\rho^{TG(\Sigma)} \\
([a]\phi)_\rho^{TG(\Sigma)} & = & \{s \in S \mid \forall s'.(L_A(s,a) = s') \supset \\
 & & \qquad (s' \in \phi_\rho^{TG(\Sigma)})\} \\
X_\rho^{TG(\Sigma)} & = & \rho(X) \\
(\mu X.\phi)_\rho^{TG(\Sigma)} & = & \cap\{E \subseteq S \mid \phi_{\rho[X/E]}^{TG(\Sigma)} \subseteq E\}
\end{array}$$

In fact, we are interested in closed formulae ϕ, whose extension in $TG(\sigma)$ is independent of the valuation: each such formula will be denoted simply by $\phi^{TG(\Sigma)}$.

Given $\phi \in \mathcal{L}$, we denote with $d(\phi)$ the formula obtained from the negation normal form of ϕ by replacing each occurrence of a subformula of the form $[a_f]\psi$, in which a_f is a sensing action symbol, with the formula $[a_f^+]\psi \wedge [a_f^-]\psi$, and replacing each occurrence of a subformula of the form $\langle a_f \rangle\psi$, in which a_f is a sensing action symbol, with the formula $\langle a_f^+\rangle\psi \wedge \langle a_f^-\rangle\psi$,

Formally the relationship between a knowledge base Σ and its transition graph is given by the following theorem.

Theorem 1 *Let Σ be a specification of a dynamic system as above, and let ϕ be any closed dynamic formula in $\mathcal{L}$. Then, $\Sigma \models \phi(s_{init})$ if and only if $s_{init} \in d(\phi)^{TG(\Sigma)}$.*

Observe that, being $TG(\Sigma)$ essentially a finite "transition system" whose nodes represent sets of valid propositional formulae, it is immediate to modify model checking algorithms for modal mu-calculus formulae for finite transition systems [24; 3; 19], to verify whether s_{init} is in the extension of a formula in $TG(\Sigma)$, and hence, by Theorem 1, to reason about actions in our setting.

5 Reasoning about actions in $\mathcal{L}$

We illustrate the how the formalism proposed can be used for various forms of reasoning about actions. Below, we informally say that a formula "holds" in a state if the formula is "known" in the robot's corresponding epistemic state.

Projection problem

We start by expressing the *projection problem*: "does a proposition p hold after the execution of a given sequence of actions, say a_1, a_2, a_3?" This can be checked by verifying the following logical implication:

$$\Sigma \models [\langle a_1\rangle\langle a_2\rangle\langle a_3\rangle \mathbf{k}p](s_{init})$$

where $\langle a_1\rangle\langle a_2\rangle\langle a_3\rangle \mathbf{k}p$ expresses that the sequence of actions a_1, a_2, a_3 can indeed be executed and that it leads to a state where p holds.

Planning

Let us now consider the *planning problem*: "is there a sequence of actions that leads to a state where a given goal p_{goal} holds?". This can be expressed by

$$\Sigma \models [\mu X.\, \mathbf{k}p_{goal} \vee \bigvee_{a\in Act} \langle a\rangle X](s_{init}) \qquad (2)$$

The dynamic formula on the right-hand side denotes the following inductive property: either p_{goal} holds in the current state, or there is an action a that leads to a state from which there exists a sequence of actions that leads to a state where p_{goal} holds. Notice that, in the presence of sensing actions, the planning process has to return a *conditional plan* [17]. In fact, the presence of sensing actions implies that if property (2) holds, then in each model of Σ there exists a sequence of actions, leading to the goal, which is different in the different models.

It can be shown that in our setting a conditional plan can be effectively returned by visiting $TG(\Sigma)$ and introducing an if-then-else statement on the sensed condition right after each sensing action. Notice that our formalization guarantees that the existence of a plan can be inferred if and only if there exists a *constructive* (conditional) plan which achieves the goal. That is, unrealizable plans are discarded a priori.

The planning problem can be more sophisticated than what shown above. For example we may want to do *planning with archiving and maintenance goals*: "is there a sequence of actions which achieves a certain goal p_{agoal} while another goal p_{mgoal} is always satisfied?". This can be expressed by modifying the formula used above as follows:

$$\mu X.\, \mathbf{k}p_{mgoal} \wedge (\mathbf{k}p_{agoal} \vee \bigvee_{a\in Act} \langle a\rangle X)$$

expressing the fact that, inductively, either both p_{mgoal} and p_{agoal} hold in the current state, or p_{mgoal} holds and there is an action a leading to a state where there exists a sequence achieving p_{agoal} while maintaining p_{mgoal}.

Safeness, invariance, and liveness

Next we consider *safeness properties.* These in general are properties that express that "something bad can never happen". For example, "it is not possible to reach a state from which there exists no plan to get the batteries charged"; in other words, in any reachable state the robot can formulate a plan to charge its battery. In $\mathcal{L}$, the existence of a plan to charge the batteries can be expressed, as shown above, by: $\phi_{pcb} \doteq \mu X.\mathbf{k}BttrChrgd \vee \bigvee_{a\in Act}\langle a\rangle X$. The fact that this can always be done (a safeness property) is expressed as

$$\nu X.\, \phi_{pcb} \wedge \bigwedge_{a\in Act} [a]X.$$

Invariance properties can be expressed in an analogous way, since they can be seen as safeness properties: the bad thing is the violation of the invariant. *Liveness properties*, that in general express that "something good is eventually achieved", can also be captured. For example, "a given job eventually comes to an end" can be expressed as

$$\mu X.\, \mathbf{k}JobEnded \vee (\bigvee_{a\in Act} \langle a\rangle \mathbf{k}tt) \wedge (\bigwedge_{a\in Act} [a]X)$$

Liveness and safeness conditions can be used together to express complex properties as "whenever a job is started, the job is also terminated":

$$\nu X.\, [startjob]\psi \wedge \bigwedge_{a\in Act} [a]X$$

where

$$\psi = \mu Y.\, (\bigvee_{a\in Act} \langle a\rangle \mathbf{k}tt) \wedge (\bigwedge_{a\in Act \wedge a\neq endjob} [a]X)$$

Observe the use of ψ to express the well-foundedness of all sequences of actions not including *endjob.*

Programs

Finally, we introduce a notion of robot program in order to enforce a control flow on actions. Robot programs are not part of the basic action theory specifying the general behavior of the robot; instead, they are used on top of the action theory to introduce a notion of control on the robot actions. This way to proceed mirrors the one used in developing GOLOG [16].

We consider a simple programming language that allows for building nondeterministic while-programs:

$$\delta \; ::= \; \texttt{nop} \mid a \mid \delta_1;\delta_2 \mid \delta_1|\delta_2 \mid \texttt{if}\ \phi\ \texttt{then}\ \delta_1\ \texttt{else}\ \delta_2 \mid \texttt{while}\ \phi\ \texttt{do}\ \delta$$

where `nop` is a special instruction that does nothing, a is the command requiring the execution of the action a, ";" is the sequential composition, "|" is nondeterministic choice, and `if · then · else ·` and `while · do ·` are the classical if-then-else and while constructs. The semantics of the various constructs is the usual one (see e.g. [21]), except for atomic actions, whose semantics is given by the basic action theory.

As in the case of GOLOG [16], formally programs are not part of the formalism $\mathcal{L}$. They are used to define suitable macros that are translated into $\mathcal{L}$ dynamic formulae.

We illustrate this approach by showing how to express the property "there exists a terminating execution of program δ that terminates in a state where ϕ holds", which corresponds to the expression $\exists s'.DO(\delta, s, s') \wedge \Phi(s)$ used in GOLOG computations [16]. We can capture the property by defining a $\mathcal{L}$ dynamic formula $\mathit{afterS}(\delta, \phi)$ by induction on the structure of the program as follows (we define $\langle \mathtt{nop} \rangle \phi = [\mathtt{nop}]\phi = \phi$):

$$
\begin{array}{l}
\mathit{afterS}(\mathtt{nop}, \phi) = \phi \\
\mathit{afterS}(a, \phi) = \langle a \rangle \phi \\
\mathit{afterS}(\delta_1; \delta_2, \phi) = \\
\qquad \mathit{afterS}(\delta_1, \mathit{afterS}(\delta_2, \phi)) \\
\mathit{afterS}(\delta_1 | \delta_2, \phi) = \\
\qquad \mathit{afterS}(\delta_1, \phi) \ \vee \ \mathit{afterS}(\delta_2, \phi) \\
\mathit{afterS}(\mathtt{if}\ \phi'\ \mathtt{then}\ \delta_1\ \mathtt{else}\ \delta_2, \phi) = \\
\qquad \phi' \wedge \mathit{afterS}(\delta_1, \phi) \ \vee \ \neg\phi' \wedge \mathit{afterS}(\delta_1, \phi) \\
\mathit{afterS}(\mathtt{while}\ \phi'\ \mathtt{do}\ \delta, \phi) = \\
\qquad \mu X. \neg\phi' \wedge \phi \ \vee \ \phi' \wedge \mathit{afterS}(\delta, X)
\end{array}
$$

Notice that the formula $\mathit{afterS}(\delta, \phi)$ is particularly meaningful if we assume that, at the various choice points of the program, the robot can do the choice, choosing the execution that eventually leads to termination in a state where ϕ holds (exactly as assumed by GOLOG computations).

The expressive abilities of $\mathcal{L}$ allow for the formalization of a wide variety of program properties. As a further example, the property "all executions of program δ terminate in states where ϕ holds", can be expressed as the $\mathcal{L}$ formula $\mathit{afterA}(\delta, \phi)$ defined as $\mathit{afterS}(\delta, \phi)$ except that the disjunction in the fourth equation is replaced by a conjunction.[1] Hence, the only difference between the definitions of $\mathit{afterA}(\delta, \phi)$ and $\mathit{afterS}(\delta, \phi)$ is in the treatment of the choice construct: in the case of $\mathit{afterA}(\delta, \phi)$ we require that, independently of the choices made, the program terminates in a state satisfying ϕ, while in the case of $\mathit{afterA}(\delta, \phi)$ only one such choice has to do so. That is, $\mathit{afterA}(\delta, \phi)$ is especially meaningful if the robot has no control on the choice points of the program, so we require that the program "does the right thing" independently of the choices made.[2] We also observe that typical *total correctness* conditions, usually written as $[\phi_1]\delta[\phi_2]$, are expressible by $\phi_1 \supset \mathit{afterA}(\delta, \phi_2)$. Instead, *partial correctness* conditions (correctness for terminating executions only), usually written as $\{\phi_1\}\delta\{\phi_2\}$, are expressible by $\phi_1 \supset \mathit{afterAw}(\delta, \phi_2)$, where $\mathit{afterAw}(\delta, \phi)$ is the formula obtained from $\mathit{afterA}(\delta, \phi)$ replacing $\langle a \rangle \phi$ in the first equation by $[a]\phi$, and the least fixpoint μ in the last equation by the greatest fixpoint ν.

[1]Observe that $\langle a \rangle \phi \equiv \langle a \rangle \mathbf{k} tt \wedge [a]\phi$, since actions are assumed to be deterministic.

[2]Notice that $\mathit{afterS}(\delta, \phi)$ is expressible in PDL (leaving aside the $\mathbf{k}$ operator), while $\mathit{afterA}(\delta, \phi)$ is not.

6 Conclusions

In this paper we have shown a way to combine model checking for a very expressive logic of programs with propositional inference, in order to exploit model checking techniques and systems for sophisticated forms of reasoning about actions, including planning and reasoning about program executions. In particular, we have applied such techniques in a framework which enables for both representing rich dynamic systems (it allows for dealing with sensing actions and incomplete information) and efficiently verifying complex properties of such systems (expressed in the language of the modal mu-calculus).

The work presented is related to several proposals in reasoning about actions. We already mentioned the connection with GOLOG [16]. Moreover, in [15] a "validity/provability based GOLOG" has been developed, which shares, in fact, some of the ideas behind our transition graph construction.

There are also some similarities with $\mathcal{A}$-like action languages (see e.g. [1; 10]; indeed, the semantics of $\mathcal{A}$ languages is based on a single transition function, and this allows for building a single transition graph. States in such graph are characterized by the formulae that are *true* (vs. *valid*), while the initial state is replaced by a set of possible initial states. Notably, model checking techniques could be adopted in that setting as well [5].

Model checking is the basic reasoning technique used in [4], where a process algebra is introduced to specify the behavior of the dynamic system, and a suitable variant of modal mu-calculus is adopted as verification formalism. Interestingly, programs (processes) in that work have a somewhat different role, since they are used for specifying basic behavior of the robot and are not considered in the verification formalism.

References

[1] C. Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *Proc. of IJCAI'93*, pages 866–871, 1993.

[2] C. Baral and T. Son. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. of ILPS-97*, 1997.

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

[4] X. J. Chen and G. De Giacomo. Reasoning about nondeterministic and concurrent actions: A process algebra approach. *Artif. Intell.*, 107:63–98, 1999.

[5] A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in nondeterministic domains. In *Proc. of AAAI'98*, pages 875–881, 1998.

[6] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Moving a robot: the KR&R approach at work. In *Proc. of KR'96*, pages 198–209, 1996.

[7] 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.

[8] F. M. Donini, D. Nardi, and R. Rosati. Autoepistemic description logics. In *Proc. of IJCAI'97*, pages 136–141, 1997.

[9] E. A. Emerson. Automated temporal reasoning about reactive systems. In *Logics for Concurrency: Structure versus Automata*, number 1043 in Lecture Notes in Computer Science, pages 41–101. Springer-Verlag, 1996.

[10] E. Giunchiglia and V. Lifschitz. An action language based on causal explanations: preliminary report. In *Proc. of AAAI'98*, 1998.

[11] K. Golden and D. Weld. Representing sensing actions: the middle ground revisited. In *Proc. of KR'96*, pages 174–185, 1996.

[12] C. Hoare. *Communicating Sequential Processes.* Prentice Hall Int., London, 1985.

[13] 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, 1990.

[14] G. Lakemeyer and H. J. Levesque. AOL: a logic of acting, sensing, knowing, and only knowing. In *Proc. of KR'98*, pages 316–327. Morgan Kaufmann, Los Altos, 1998.

[15] Y. Lesperance and D. Tremaine. A procedural approach to belief update for agent programming. Unpublished Manuscript, Department of Computer Science York University, 1998.

[16] H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[17] H. J. Levesque. What is planning in presence of sensing? In *Proc. of AAAI'96*, pages 1139–1149. AAAI Press/The MIT Press, 1996.

[18] J. Lobo, G. Mendez, and S. R. Taylor. Adding knowledge to the action description language A. In *Proc. of AAAI'97*, pages 454–459, 1997.

[19] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[20] M. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[21] H. R. Nielson and F. Nielson. *Semantics with Applications.* Wiley, 1992.

[22] R. Reiter. *Knowledge in Action: Logical Foundation for Describing and Implementing Dynamical Systems.* 1998. In preparation.

[23] M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* The MIT Press, 1997.

[24] C. Stirling. Modal and temporal logics for processes. In *Logics for Concurrency: Structure versus Automata*, number 1043 in Lecture Notes in Computer Science, pages 149–237. Springer-Verlag, 1996.