



Programming rational agents in a modal action logic

Matteo Baldoni^a, Alberto Martelli^a, Viviana Patti^a and Laura Giordano^b

^a *Dipartimento di Informatica, Università degli Studi di Torino, C.so Svizzera 185, I-10149 Torino, Italy*
E-mail: {baldoni,mrt,patti}@di.unito.it

^b *Dipartimento di Informatica, Università degli Studi del Piemonte Orientale, Spalto Marengo, 33, I-15100 Alessandria, Italy*
E-mail: laura@di.unito.it

In this paper we describe a language for reasoning about actions that can be used for modelling and for programming rational agents. We propose a modal approach for reasoning about dynamic domains in a logic programming setting. Agent behavior is specified by means of *complex actions* which are defined using *modal inclusion axioms*. The language is able to handle knowledge producing actions as well as actions which remove information. The problem of reasoning about complex actions with incomplete knowledge is tackled and the temporal projection and planning problems is addressed; more specifically, a *goal directed proof procedure* is defined, which allows agents to reason about complex actions and to generate conditional plans. We give a *non-monotonic* solution for the frame problem by making use of persistency assumptions in the context of an abductive characterization. The language has been used for implementing an adaptive web-based system.

Keywords: logic-based agents, modal and multimodal logic, logic programming, reasoning with incomplete knowledge, reasoning about actions

1. Introduction

In this paper we describe a language for reasoning about actions that can be used for specifying agents and for executing agent specifications. Reasoning about the effects of actions in a dynamically changing world is one of the main problems which must be faced by intelligent agents and, on the other hand, the internal dynamics of the agent itself can be regarded as resulting from the execution of actions on the mental state.

The action theory we adopt is a modal action theory, in which actions are represented by modalities. The adoption of Dynamic Logic or a modal logic to deal with the problem of reasoning about actions and change is common to many proposals, as for instance [16,19,29,38,41], and it is motivated by the fact that modal logic allows a very natural representation of actions as state transitions, through the accessibility relation of Kripke structures. Since the intentional notions (or attitudes), which are used to describe agents, are usually represented as modalities, our modal action theory is also well suited to incorporate such attitudes, although in this paper we will only deal with beliefs.

Our starting point is the modal logic programming language for reasoning about actions presented in [12]. Such language extends the language \mathcal{A} [27] to deal with ram-

ifications, by means of “causal rules” among fluents, and with nondeterministic actions. The language relies on an abductive semantics, to provide a nonmonotonic solution to the frame problem, and, when there are no ramifications, it has been proved to be equivalent to the language \mathcal{A} .

Such a language mainly focuses on *ramification problem* but does not provide a formalization of incomplete initial states with an explicit representation of *undefined* fluents. Such an explicit representation is needed if we want to model an agent which is capable of reasoning and acting on the basis of its (dis)beliefs. Often an agent has incomplete information about the state of the world and it needs to take actions to obtain the knowledge necessary for determining how to act. These knowledge producing actions [40] are usually called *sensing* actions.

There are several proposals in the literature for reasoning about actions in the presence of sensing which have been developed along the line of a Scherl and Levesque paper [40]. Let us mention the work on the high level robot programming language GOLOG [20,21], which is based on a theory of actions in the situation calculus. Other proposals have been developed by extending the action description language \mathcal{A} [27], as in [14,35], while in [42] a formal account of a robot’s knowledge about the state of its environment has been developed in the context of the fluent calculus.

In this paper, we start from the action language presented in [12] to define a language capable of representing *incomplete belief states* and of dealing with *sensing actions* as well as with *complex actions*. Our aim is that of defining an agent programming language in which the possible behaviors of an intelligent agent reasoning and acting in the world can be described. In particular, the agent should be able to plan its behavior by choosing a given course of actions among different possible ones according to its current beliefs on the world, and can use sensors for acquiring the information it lacks about the real world. As we will focus on modeling a single agent and its interactions with the world, we will be only concerned with the description of the internal dynamics of the agent, which results from the effects of actions execution on its mental state: actions have certain effects on the world and the agent updates its belief accordingly, when executing an action. Moreover the agent can check the current state of the world, on which he might have an incomplete information or which might have been modified by external (exogenous) events, by making use of sensing actions. For these reasons, in our formalism we will only keep the agent’s representation of the world, while in other formalizations of sensing actions [14,40], where the focus is on developing a theory of actions and knowledge rather than on modeling agent behaviors, both the mental state of the agent and the real state of the world are represented. In order to represent the mental state of an agent, we introduce an epistemic level in our action logic. By using belief modalities, we represent the mental state of an agent as a set of epistemic literals. Then, as concerns world actions, i.e. actions affecting the real world, we only model what the agent believes about action’s effects based on its beliefs about the preconditions. As concerns sensing actions, we consider them as input actions which produce fresh information on the value of some fluents in the real world. In essence, sensing actions are regarded as non-deterministic actions, whose outcome cannot be predicted by the agent.

Another aim of the paper is to extend the action language to deal with *complex actions*. The definition of complex actions we introduce draws from dynamic logic [30] for the definition of action operators like sequence, test and non-deterministic choice. However, rather than referring to an Algol-like paradigm for describing complex actions, as in [34], we refer to a Prolog-like paradigm: complex actions are defined through (possibly recursive) definitions, given by means of Prolog-like clauses.

In particular, we show that in modal logics, we can express complex action definitions by means of a suitable set of axioms of the form

$$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle \varphi.$$

If p_0 is a procedure name, and the p_i ($i = 1, \dots, n$) are either procedure names, or atomic actions or tests, the above axiom can be interpreted as a procedure definition, which can then be executed in a goal directed way, similarly to standard logic programs. These axioms have the form of inclusion axioms, which were the subject of a previous work [3,11], in which the class of multimodal logics characterized by axioms of the form $[s_1] \dots [s_m] \varphi \subset [p_1] \dots [p_n] \varphi$, where $[s_i]$ and $[p_i]$ are modal operators, have been analyzed. These axioms have interesting computational properties because they can be considered as rewriting rules.

This action theory has an immediate computational counterpart in a logic programming language, called DyLOG. We develop a goal directed proof procedure for reasoning about complex actions (including sensing actions), which can be considered as an interpreter of such a language. This procedure can be extended for constructing *conditional plans* to achieve a given goal from an incompletely specified initial state. We can prove in the language that such generated plans are *correct*, i.e. achieve the desired goal for a given initial state. DyLOG can be used as an ordinary programming language for executing procedures which model the behavior of an agent, but also to reason about them, by extracting from them linear or conditional plans. It has been implemented in Sictus Prolog. Its interpreter is a straightforward implementation of its operational semantics and can be downloaded in [1]. DyLOG has been practically used for specifying the behavior of rational agents that supply adaptive services in a web-based application context [4,9].

The outline of the paper is as follows. In section 2 the modal action logic is presented through an example. In section 3 we describe the axiomatics and the Kripke semantics of the action logic and, also, we describe the abductive semantics on which is based our solution to the frame problem. Section 4 describes a goal directed proof procedure for computing linear plans to achieve a given goal, as well as a procedure which computes conditional plans. Section 5 describes the use of this agent programming language to implement adaptive web applications. Sections 6 and 7 contain the discussion of related work and some conclusion.

2. The modal action logic

In our action logic actions are represented by a modality. We distinguish between three kinds of actions: *sensing actions*, which affect the internal state of the agent by

enhancing its knowledge on the environment, *world actions*, that is actions which have actual effects on the external world, and *complex actions*, which are defined on the base of the previous ones together with test and possibly complex actions as well. We denote by \mathcal{A} the set of world actions and by \mathcal{S} the set of sensing actions. For each action $a \in \mathcal{A} \cup \mathcal{S}$ we introduce a normal modal operator $[a]$ ¹. A formula $[a]\alpha$ means that α holds after every execution of action a , while $\langle a \rangle \alpha$, the dual of $[a]$, means that there is a possible execution of action a after which α holds (and similarly for the modalities for sensing actions). Furthermore, we make use of the normal modal operator \Box to denote *laws*, namely formulas that hold *always*, after any sequence of actions. The intended meaning of a formula $\Box\alpha$ is that α holds after every sequence of actions. The modality \Box is ruled by the axioms of the logic *S4* (reflexivity and transitivity) and it interacts with the world actions modalities through the axiom schema $\Box\varphi \supset [a]\varphi$ for each $a \in \mathcal{A}$. This is sufficient to guarantee that by \Box we can denote a (possibly empty) arbitrary sequence of actions. In fact by reflexivity ($\Box\varphi \supset \varphi$), a formula that holds always will also hold in an initial moment where no action has been executed yet; by transitivity ($\Box\varphi \supset \Box\Box\varphi$) and by the interaction axiom $\Box\varphi \supset [a]\varphi$, we have that a formula that holds always will also hold after an arbitrary sequence of world actions. In order to represent complex actions, the language contains also a finite number of normal modal operators $[p_i]$, where p_i is a constant denoting a procedure name. We will use the modality $\langle p_i \rangle$ as the dual of $[p_i]$ and we denote by \mathcal{P} the set of such procedure names.

In our language we use atomic propositions for *fluent names*, and l to denote a *fluent literal*, consisting in a fluent name f or its negation $\neg f$. Since we want to reason about the effects of actions on the internal state of an agent, we define a state as a set of belief formulas. More precisely, we introduce the notion of *epistemic fluent literal* to be a modal atom $\mathcal{B}l$ or its negation $\neg\mathcal{B}l$, where l is a fluent literal and the modal operator \mathcal{B} is used to model agent's beliefs. Intuitively, $\mathcal{B}l$ means that l is believed to be the case. Moreover, we use the modality \mathcal{M} , which is defined as the dual of \mathcal{B} , i.e. $\mathcal{M}l$ is $\neg\mathcal{B}\neg l$. $\mathcal{M}\alpha$ means that α is considered to be possible. The modal operator \mathcal{B} is of type *KD* and it is a normal modality ruled by the axiom schema $\mathcal{B}\varphi \supset \neg\mathcal{B}\neg\varphi$ (seriality). It is worth noting that the usual modal logic used to represent belief operators is *KD45*. In our formalization we do not add the positive and negative introspection axioms to belief modality \mathcal{B} because we restrict it to be used in front of literals. Intuitively, this means that the epistemic state of an agent is restricted to contain beliefs on the facts of the world and not on other beliefs.²

Definition 2.1 (Epistemic state). An *epistemic state* S is a set of epistemic literals satisfying the following requirements:

1. for each fluent literal l , either $\mathcal{B}l \in S$ or $\neg\mathcal{B}l \in S$ (completeness);
2. for no fluent literal l , both $\mathcal{B}l \in S$ and $\neg\mathcal{B}l \in S$ (consistency).

¹ That is, the modal operator $[a]$ is ruled at least by axiom *K* ($[a]$): $[a](\alpha \supset \beta) \supset ([a]\alpha \supset [a]\beta)$.

² In [7] we extend the framework in order to deal with epistemic states containing nested beliefs of rank two. In that context \mathcal{B} is ruled by axioms of the logic *KD45*.

Notice that, since \mathcal{B} is serial the case when both $\mathcal{B}l$ and $\mathcal{B}\neg l$ hold cannot occur. In fact, if $\mathcal{B}l \in S$ then, by seriality, $\neg\mathcal{B}\neg l \in S$ and thus, by consistency, $\mathcal{B}\neg l \notin S$. Similarly when $\mathcal{B}\neg l \in S$. In essence a state is a complete and consistent set of epistemic fluent literals, and it provides a *three-valued* interpretation in which each literal l is *true* when $\mathcal{B}l$ holds, *false* when $\mathcal{B}\neg l$ holds, and *undefined* when both $\neg\mathcal{B}l$ and $\neg\mathcal{B}\neg l$ hold (denoted by $\mathcal{U}l$).

Finally, the operators “ \cup ” (*non-deterministic choice*), “ $;$ ” (*sequence*) and “ $?$ ” (*test*) of dynamic logic [30] are used for building complex actions from world actions, sensing actions, and complex action as well. They are ruled by the axiom schemas: $\langle a \cup b \rangle \varphi \equiv \langle a \rangle \varphi \vee \langle b \rangle \varphi$, $\langle a; b \rangle \varphi \equiv \langle a \rangle \langle b \rangle \varphi$, and $\langle \psi? \rangle \varphi \equiv \psi \wedge \varphi$. The operator “ \cup ” expresses the non-deterministic choice among two actions: executing the choice $a \cup b$ means to execute non-deterministically either a or b . Instead, the operator “ $;$ ” expresses the sequencing of two actions: executing $a; b$ means to execute the actions a and b in this order. Finally, for the test operator, similarly to dynamic logic, if ψ is a formula then $\psi?$ can be used as a label for a modal operator, such as $\langle \psi? \rangle$. However, in our language, we restrict the formulae that can occur as a label of a test operator to a conjunction of epistemic fluents.

2.1. World actions

World actions allow the agent to affect the environment. In our formalization we model the epistemic state of the agent while we do not model the real world. For this reason we will not represent the actual effects of actions on the world and we only formalize what the agent believes about these effects based on belief preconditions. For each world action, the domain description contains a set of *simple action clauses*, that allow direct effects and preconditions of world actions on the epistemic state to be described. Basically, simple action clauses consist of *action laws* and *precondition laws*.³

Action laws define direct effects of a world actions on an epistemic fluent and allow actions with conditional effects to be represented. They have form:

$$\Box(\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n \supset [a]\mathcal{B}l_0), \quad (1)$$

$$\Box(\mathcal{M}l_1 \wedge \dots \wedge \mathcal{M}l_n \supset [a]\mathcal{M}l_0). \quad (2)$$

(1) means that after any sequence of world actions (\Box), if the set of fluent literals l_1, \dots, l_n (representing the preconditions of the action a) is believed then, after the execution of a , l_0 (the effect of a) is also believed. (2) is necessary in order to deal with *ignorance* about preconditions of the action a . It means that the execution of a may affect the beliefs about l_0 , when executed in a state in which the preconditions are considered to be possible. When the preconditions of a are unknown, this law allows to conclude that the effects of a are unknown as well.

³ In this paper we do not introduce constraints or causal rules among fluents. However, causal rules could be easily introduced by allowing a causality operator, as in [29] to which we refer for a treatment of ramification in a modal setting.

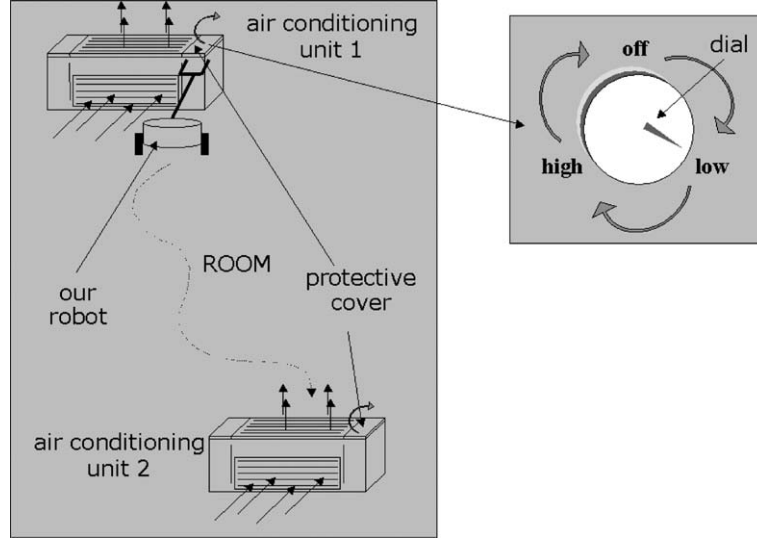


Figure 1. A snapshot of our robot. Initially it is inside the room, in front of the air conditioning unit number 1.

Example 2.1. Let us consider the example of a robot which is inside a room. Two air conditioning units, *unit1* and *unit2* can blow fresh air in the room and the flow of the air from a unit can be controlled by a dial as the one represented in figure 1. Such kind of dial can be accessed by raising a protective cover and can be set in three positions: off, low, high. $turn_dial(I)$ denotes the atomic action of turning the dial controlling the unit I clockwise from a position to the next one, which causes to start the air flow blowing with low speed when the unit is off, to increase the speed of the flow if it is low, and, finally, to stop the unit's blowing if the flow is high. Note that, in order to avoid the introduction of many variant of the same clauses, as a shorthand, we use the metavariable I , where $I \in \{unit1, unit2\}$ and the metavariable P , where $P \in \{low, high, off\}$, in other words $flow(I, P)$ can be regarded as an atomic proposition.

- (a) $\Box(\mathcal{B}flow(I, low) \supset [turn_dial(I)]\mathcal{B}flow(I, high)),$
- (b) $\Box(\mathcal{M}flow(I, low) \supset [turn_dial(I)]\mathcal{M}flow(I, high)),$
- (c) $\Box(\mathcal{B}flow(I, high) \supset [turn_dial(I)]\mathcal{B}flow(I, off)),$
- (d) $\Box(\mathcal{M}flow(I, high) \supset [turn_dial(I)]\mathcal{M}flow(I, off)),$
- (e) $\Box(\mathcal{B}flow(I, off) \supset [turn_dial(I)]\mathcal{B}flow(I, low)),$
- (f) $\Box(\mathcal{M}flow(I, off) \supset [turn_dial(I)]\mathcal{M}flow(I, low)).$

The above action laws describe some of the effects of turning a dial on the agent's epistemic state: depending on its beliefs about the position of the dial before executing the action, the agent gains knowledge about the new dial's position. Moreover, since in

this paper we do not treat causal rules and constraints, some action law must be introduced in order to describe the fact that, as a consequence of turning the dial on the next position, the agent comes to know that the dial is not in the previous position anymore⁴:

$$(g) \quad \Box(\mathcal{B}flow(I, P) \supset [turn_dial(I)]\mathcal{B}\neg flow(I, P)),$$

$$(h) \quad \Box(\mathcal{M}flow(I, P) \supset [turn_dial(I)]\mathcal{M}\neg flow(I, P)).$$

Precondition laws allow to specify belief preconditions for world actions, that is those epistemic conditions which make an action executable in a state. They have form:

$$\Box(\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n \supset \langle a \rangle \top) \quad (3)$$

meaning that after any sequence of world actions, if the conjunction of epistemic literals $\mathcal{B}l_1, \dots, \mathcal{B}l_n$ holds, then a can be executed. For instance, according to the following clause, the robot must know to be in front of the air conditioning unit I and the protective cover to be raised if it wants to turn the dial controlling the unit's flow by executing $turn_dial(I)$:

$$(i) \quad \Box(\mathcal{B}in_front_of(I) \wedge \mathcal{B}cover_up(I) \supset \langle turn_dial(I) \rangle \top)$$

2.1.1. Knowledge removing actions

Up to now, we considered actions with deterministic effects on the world, i.e. actions in which the outcome can be predicted. The execution of such actions causes the agent to have information about their effects, because the action is said to *deterministically* cause the change of a given set of fluents. However effects of actions can be non-deterministic and, then, unpredictable. In such a case, the execution of the action causes the agent to *lose knowledge* about its possible effects, because the action could unpredictably cause the change of some fluent. In our framework, we can model actions with non-deterministic effects as actions which *may* affect the beliefs about the value of a fluent, by simply using action laws of form (2) but without adding the corresponding law of the form (1).

Example 2.2. Let us consider an action *drop* of dropping a glass from a table. We want to model the fact that dropping a *fragile* glass may possibly make the glass broken. It can be expressed by using a suitable action law of the form (2):

$$\Box(\mathcal{M}fragile \supset [drop(I)]\mathcal{M}broken).$$

It means that, in the case the agent considers possible that the glass is fragile, then, after dropping it, it considers possible that it has become broken. Note that, since $\mathcal{B}\alpha$ entails $\mathcal{M}\alpha$ (seriality), the action law above can also be applied in the case the agent believes that the glass is fragile, to conclude that it is possibly broken. If action *drop* is

⁴ If causal laws would be available, this could be expressed as an indirect effect of the action of turning, by means of a causal law defining a dependency relationship between the value of the fluents representing the three different dial positions.

executed in a state in which $\mathcal{B}fragile$ and $\mathcal{B}\neg broken$ hold, in the resulting state $\mathcal{M}broken$ (i.e. $\neg\mathcal{B}\neg broken$) will hold: the agent does not know anymore if the glass is broken or not (for a formal account of fluents' persistency after executing actions, see section 3).

2.2. Sensing actions

Let us now consider sensing actions, which allow an agent to *gather information from the environment*, enhancing its knowledge about the value of a fluent. In modelling sensing actions we are interested to represent the possible epistemic states achieved by the agent after the execution of sensing. In fact, since the actual result of the sensing execution cannot be predicted, in order to represent the effects of such actions on the epistemic state we have to take into account all the possibilities. Let us consider the case of binary sensing actions, that allow to gather information about a fluent l . In this case the sensing can have two possible outcomes, the one informing that l is true, the other informing the it is false; thus there are two possible epistemic state resulting from the execution of the action, the first containing the belief that l holds, the second containing the belief that l do not hold. In order to formalize it in our framework, we define a sensing action as a non-deterministic action. In particular, a *binary* sensing action $s \in \mathcal{S}$, that allow to check whether the fluent l or its complement $\neg l$ is true, is represented by means of axioms of our logic that specify the effects of s on agent beliefs as the non-deterministic choice between two *ad hoc* world actions $s^{\mathcal{B}l}$ and $s^{\mathcal{B}\neg l}$: the former causing the belief $\mathcal{B}l$, and the latter causing the belief $\mathcal{B}\neg l$.

The intuition is that from the point of view of the agent a sensing action on a fluent literal l has the non-deterministic effect of making l to be believed or $\neg l$ to be believed. Thus, for each binary sensing action $s \in \mathcal{S}$ we have an axiom of form:

$$[s]\varphi \equiv [s^{\mathcal{B}l} \cup s^{\mathcal{B}\neg l}]\varphi$$

which expresses the non-deterministic choice (by means of the choice operator of dynamic logic "U") among the two world actions $s^{\mathcal{B}l}$ and $s^{\mathcal{B}\neg l}$. Conditions which make the sensing action s executable in a epistemic state are represented by the same preconditions of the defining actions $s^{\mathcal{B}l}$ and $s^{\mathcal{B}\neg l}$. The actions $s^{\mathcal{B}l}$ and $s^{\mathcal{B}\neg l}$ can be regarded as being predefined actions ruled by the following simple action clauses:

$$\begin{aligned} \Box(\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n \supset \langle s^{\mathcal{B}l} \rangle \top), & \quad \Box(\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n \supset \langle s^{\mathcal{B}\neg l} \rangle \top), \\ \Box(\top \supset [s^{\mathcal{B}l}] \mathcal{B}l), & \quad \Box(\top \supset [s^{\mathcal{B}\neg l}] \mathcal{B}\neg l). \end{aligned}$$

Summarizing, the formulation above expresses the fact that s can be executed in a state where the preconditions $\mathcal{B}l_1, \dots, \mathcal{B}l_n$ hold, leading to a new state where the agent has acquired a belief about l : he may either believe that l or that $\neg l$, see figure 2.

Example 2.3. Let $sense_cover(I) \in \mathcal{S}$ denote the action of sensing whether the cover protecting the dial that controls a unit I is raised, which is executable if the robot believes to be in front of the unit I . This is the suitable axiom representing belief precondition and effects:

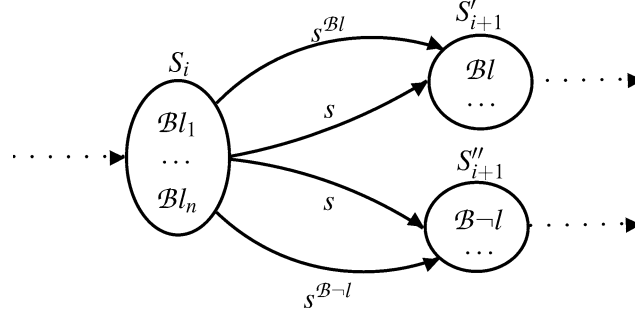


Figure 2. The execution of a sensing action on a fluent l leads to a new state where the agent has acquired a belief about l : either $\mathcal{B}l$ or $\mathcal{B}\neg l$.

$$(f) [\text{sense_cover}(I)]\varphi \equiv [\text{sense_cover}(I)^{\mathcal{B}\text{cover_up}(I)} \cup \text{sense_cover}(I)^{\mathcal{B}\neg\text{cover_up}(I)}]\varphi$$

where the sensing actions $\text{sense_cover}(I)^{\mathcal{B}\text{cover_up}(I)}$ and $\text{sense_cover}(I)^{\mathcal{B}\neg\text{cover_up}(I)}$ are ruled by the set of laws:

- (j) $\Box(\mathcal{B}\text{in_front_of}(I) \supset \langle \text{sense_cover}(I)^{\mathcal{B}\text{cover_up}(I)} \rangle \top)$,
- (k) $\Box(\top \supset [\text{sense_cover}(I)^{\mathcal{B}\text{cover_up}(I)}]\mathcal{B}\text{cover_up}(I))$,
- (l) $\Box(\mathcal{B}\text{in_front_of}(I) \supset \langle \text{sense_cover}(I)^{\mathcal{B}\neg\text{cover_up}(I)} \rangle \top)$,
- (m) $\Box(\top \supset [\text{sense_cover}(I)^{\mathcal{B}\neg\text{cover_up}(I)}]\mathcal{B}\neg\text{cover_up}(I))$.

More in general, we can deal with sensing on a *finite set of literals*, where executing a sensing action leads to a new state where the agent knows which literal is true among an associated set of literals. More formally, we associate to each sensing action $s \in \mathcal{S}$ a set $\text{dom}(s)$ of literals. The effect of s will be to have the information about which literal in $\text{dom}(s)$ is true. This is modeled by introducing an axiom of the form:

$$[s]\varphi \equiv \left[\bigcup_{l \in \text{dom}(s)} s^{\mathcal{B}l} \right] \varphi \quad (4)$$

where for each $l \in \text{dom}(s)$ the sensing action $s^{\mathcal{B}l} \in \mathcal{A}$ is ruled by the following simple action clauses:

$$\Box(\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n \supset \langle s^{\mathcal{B}l} \rangle \top), \quad (5)$$

$$\Box(\top \supset [s^{\mathcal{B}l}]\mathcal{B}l), \quad (6)$$

$$\Box(\top \supset [s^{\mathcal{B}l}]\mathcal{B}\neg l') \quad (7)$$

for each $l' \in \text{dom}(s)$, $l \neq l'$. Clause (5) means that after any sequence of world actions, if the set of literals $\mathcal{B}l_1 \wedge \dots \wedge \mathcal{B}l_n$ holds, then the action $s^{\mathcal{B}l}$ can be executed. The other ones describe the effects of $s^{\mathcal{B}l}$: in any state, after the execution of $s^{\mathcal{B}l}$, l is believed (6), while all the other fluents belonging to $\text{dom}(s)$ are believed to be false (7). Note that the

binary sensing action on a fluent I , is a special case of sensing where the associated finite set is $\{I, \neg I\}$.

Example 2.4. Let $sense_dial(I) \in \mathcal{S}$ denotes the action of sensing the position of the dial controlling a unit I . It returns as a value one element of the associated set of literals $dom(sense_dial(I)) = \{flow(I, off), flow(I, low), flow(I, high)\}$ and it is executable if the robot believes the cover protecting the dial to be open and to be in front of the unit. This is the suitable axiom representing belief preconditions and effects:

$$(n) [sense_dial(I)]\varphi \equiv \left[\bigcup_{I \in \{flow(I, off), flow(I, low), flow(I, high)\}} sense_dial(I)^{B_I} \right] \varphi$$

where the world actions $sense_dial(I)^{B_{flow(I, off)}}$, $sense_dial(I)^{B_{flow(I, low)}}$ and $sense_dial(I)^{B_{flow(I, high)}}$ are ruled by a suitable set of simple action clauses:

- (o) $\Box (Bin_front_of(I) \wedge cover_up(I) \supset \langle sense_dial(I)^{B_{flow(I, P)}} \rangle \top)$,
- (p) $\Box (\top \supset [sense_dial(I)^{B_{flow(I, P)}}] B_{flow(I, P)})$,
- (q) $\Box (\top \supset [sense_dial(I)^{B_{flow(I, P)}}] B_{\neg flow(I, Q)})$,
- (r) $\Box (\top \supset [sense_dial(I)^{B_{flow(I, P)}}] B_{\neg flow(I, R)})$.

As before, metavariables are introduced in order to avoid to introduce many variant of the same clause. In particular we used the metavariable I , where $I \in \{unit1, unit2\}$, and the metavariables P, Q, R , where $P, Q, R \in \{off, low, high\}$ and $P \neq Q \neq R$.

2.3. Complex actions

In our modal action theory, complex actions are defined on the basis of world actions, tests, and other complex actions. A *complex action* is defined by means of a suitable set of inclusion axiom schemas of our modal logic, having the form⁵:

$$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle \varphi \quad (8)$$

where p_0 is a procedure name in \mathcal{P} , and p_i ($i = 1, \dots, n$) are either procedure names, or world actions, or sensing actions, or test. *Test actions* are needed for testing if some fluent holds in the current state and for expressing conditional complex actions. A set of axioms of form (8) can be interpreted as a *procedure definition*. Procedure definitions can be *recursive* and they can also be *non-deterministic*, when they are defined by a collection of axioms of the form specified above. Intuitively, they can be executed in a goal directed way, similarly to standard logic programs. Indeed the meaning of (8) is that if in a state there is a possible execution of p_1 , followed by an execution of p_2 , and so on up to p_n , then in that state there is a possible execution of p_0 .

Complex actions can be used to describe the behavior of an agent, as shown in the following example.

⁵ For sake of brevity, sometimes we will write these axioms as $\langle p_0 \rangle \varphi \subset \langle p_1; p_2; \dots; p_n \rangle \varphi$.

Example 2.5. Let us suppose that our robot has to achieve the goal of setting high the fan speed of an air conditioning unit U of the room (see figure 1). We define the procedure $set_the_fan_speed_high(I)$, i.e. the procedure specifying the action plans the robot may execute for achieving the goal, by means of the set of axiom schemas (A)–(F):

- (A) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset \langle \mathcal{B}flow(I, high) \rangle \varphi$,
- (B) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset$
 $\langle (\mathcal{B}in_front_of(I) \wedge \mathcal{B}cover_up(I) \wedge \mathcal{B}\neg flow(I, high)) \rangle ?$
 $\langle turn_dial(I) \rangle \langle set_the_fan_speed_high(I) \rangle \varphi$,
- (C) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset$
 $\langle (\mathcal{B}in_front_of(I) \wedge \mathcal{B}\neg cover_up(I) \wedge \mathcal{M}\neg flow(I, high)) \rangle ?$
 $\langle raise_cover(I) \rangle \langle set_the_fan_speed_high(I) \rangle \varphi$,
- (D) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset$
 $\langle (\mathcal{B}in_front_of(I) \wedge \mathcal{U}cover_up(I) \wedge \mathcal{M}\neg flow(I, high)) \rangle ?$
 $\langle sense_cover(I) \rangle \langle set_the_fan_speed_high(I) \rangle \varphi$,
- (E) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset$
 $\langle (\mathcal{B}in_front_of(I) \wedge \mathcal{B}cover_up(I) \wedge \mathcal{U}flow(I, high)) \rangle ?$
 $\langle sense_dial(I) \rangle \langle set_the_fan_speed_high(I) \rangle \varphi$,
- (F) $\langle set_the_fan_speed_high(I) \rangle \varphi \subset$
 $\langle (\mathcal{B}\neg in_front_of(I) \wedge \mathcal{M}\neg flow(I, high)) \rangle ?$
 $\langle go_to_unit(I) \rangle \langle set_the_fan_speed_high(I) \rangle \varphi$.

The definition of $set_the_fan_speed_high(I)$ is recursive and notice that it is deterministic since all clauses are mutually exclusive. The complex behavior defined is based on the world actions $turn_dial(I)$, $go_to_unit(I)$, $raise_cover(I)$ and on the sensing actions $sense_cover(I)$ and $sense_dial(I)$. Action $turn_dial(I)$ is ruled by the action laws (a)–(h) in example 2.1, and by precondition law (i) above. $sense_cover(I)$ is ruled by axiom (f) and by laws (j)–(m) in example 2.3, while $sense_dial(I)$ with $dom(sense_dial(I)) = \{flow(I, off), flow(I, low), flow(I, high)\}$, is ruled by the laws in example 2.4. The simple action clauses for the atomic actions $go_to_unit(I)$ and $raise_cover(I)$ are given as follows:

- (s) $\Box(\mathcal{B}\neg in_front_of(I) \wedge \mathcal{B}\neg out_room \supset \langle go_to_unit(I) \rangle \top)$,
- (t) $\Box(\top \supset [go_to_unit(I)]\mathcal{B}in_front_of(I))$,
- (u) $\Box(\mathcal{B}in_front_of(J) \supset [go_to_unit(I)]\mathcal{B}\neg in_front_of(J))$,
- (v) $\Box(\mathcal{M}in_front_of(J) \supset [go_to_unit(I)]\mathcal{M}\neg in_front_of(J))$,
- (w) $\Box(\mathcal{B}in_front_of(I) \supset \langle raise_cover(I) \rangle \top)$,
- (x) $\Box(\mathcal{B}\neg cover_up \supset [raise_cover(I)]\mathcal{B}cover_up(I))$,
- (y) $\Box(\mathcal{M}\neg cover_up(I) \supset [raise_cover(I)]\mathcal{M}cover_up(I))$,

where I, J are metavariables s.t. $I, J \in \{unit1, unit2\}$ and $I \neq J$. Now we can define all_units_high , which builds upon $set_the_fan_speed_high(I)$ and specifies how to achieve the goal of setting high all the air conditioning units, assuming the robot to be initially inside the room.

$$(G) \langle all_units_high \rangle \varphi \subset \langle set_the_fan_speed_high(unit1) \rangle \langle set_the_fan_speed_high(unit2) \rangle \varphi,$$

$$(H) \langle all_units_high \rangle \varphi \subset \langle set_the_fan_speed_high(unit2) \rangle \langle set_the_fan_speed_high(unit1) \rangle \varphi.$$

Notice that the two axioms defining the procedure all_units_high are not mutually exclusive: the units can be set on the value *high* in any order. The axioms specify *alternative* recipes that the robot can follow to increase the fan speed of all the room's units, each of them leading the robot to reach a different place in the room at the end of the task.

2.4. Reasoning on dynamic domain descriptions

In general, a particular dynamic domain will be described in terms of suitable simple action clauses describing precondition and effects of world actions, axioms describing the sensing and complex actions, and a set of epistemic fluents describing the initial epistemic state.

Definition 2.2 (Dynamic domain description). Given a set \mathcal{A} of world actions, a set \mathcal{S} of sensing actions, and a set \mathcal{P} of procedure names, let $\Pi_{\mathcal{A}}$ be a set of simple action clauses for world actions, $\Pi_{\mathcal{S}}$ a set of axioms of form (4) for sensing actions, $\Pi_{\mathcal{P}}$ a set of axioms of form (8). A *dynamic domain description* is a pair (Π, S_0) , where Π is the tuple $(\Pi_{\mathcal{A}}, \Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$ and S_0 is a consistent and complete set of epistemic fluent literals representing the beliefs of the agent in the initial state.

Note that $\Pi_{\mathcal{A}}$ contains also the simple actions clauses for the world actions s^{B_i} 's occurring in the axioms for sensing actions.

Example 2.6. An example of domain description is obtained by taking as $\Pi_{\mathcal{A}}$ the set of simple action clauses in examples 2.1, 2.3, 2.4 and 2.5 plus the formula (e), as $\Pi_{\mathcal{S}}$ the axiom (f) in example 2.3 and as $\Pi_{\mathcal{P}}$ the set of procedure axioms (k)–(n), (s)–(t) in example 2.5. One possible initial set of beliefs is given by state s containing the following epistemic literals: $Bin_front_of(unit1)$, $B\neg in_front_of(unit2)$, $B\neg out_room$, $Bcover_up(unit1)$, $Bflow(unit1, low)$, $B\neg flow(unit1, off)$, $B\neg flow(unit1, high)$, $Ucover_up(unit2)$, $Uflow(unit2, off)$, $Uflow(unit2, low)$, $Uflow(unit2, high)$ as well as the epistemic literals derived by seriality of the belief operators: $\neg B\neg in_front_of(unit1)$, $\neg Bin_front_of(unit2)$, $\neg Bout_room$, $\neg B\neg cover_up(unit1)$, $\neg B\neg flow(unit1, low)$, $\neg Bflow(unit1, off)$, $\neg Bflow(unit1, high)$.

Given a domain description, we can formalize two well known form of reasoning about actions: the *temporal projection* problem, where the reasoning task is to predict

the future effects of actions on the basis of (possibly incomplete) information on preceding states, and the *planning problem*, where the task is to find a sequence of actions that is *legal* (each action is executed in a context where its preconditions are satisfied) and that achieves the goal (a formula representing the goal holds in the final state that results from executing the action sequence) [33]. In particular, we formalize the *temporal projection problem* “given a sequence a_1, \dots, a_n of world actions, does the condition Fs hold after executing the actions sequence starting from the initial state?” by the query $\langle a_1 \rangle \dots \langle a_n \rangle Fs$ ($n \geq 0$), where Fs is a conjunction of epistemic literals. Notice that, since world actions $a \in \mathcal{A}$ defined in our domain descriptions are *deterministic* w.r.t. the epistemic state, as stated by proposition 4.2 in section 4.1, the equivalence $\langle a \rangle Fs \equiv [a]Fs \wedge \langle a \rangle \top$ holds for the world actions a defined in the domain description, and then, the success of the existential query $\langle a_1 \rangle \dots \langle a_n \rangle Fs$ entails the success of the universal query $[a_1] \dots [a_n]Fs$.

Moreover, we can deal with a special case of planning. Let us considering a generalization of the query above where atomic actions a_1, \dots, a_n are replaced with complex actions p_1, p_2, \dots, p_n . We get the following query:

$$\langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle Fs \quad (n \geq 0) \quad (9)$$

where $p_i, i = 1, \dots, n$, is either a world action, or a sensing action, or a procedure name, or a test. If $n = 0$ we simply write the above goal as Fs .

Intuitively, when we are faced with a query $\langle p \rangle Fs$ we ask: “is there a legal sequence of actions conforming to p that, when executed from the initial state, leads to a final state where Fs holds?”. In fact query (9) succeeds if it is possible to find a (terminating) execution of p_1, p_2, \dots, p_n (in the given order) leading to a state where Fs holds. Such terminating execution will be a *legal* sequence of deterministic (world) actions (see propositions 4.2 and 4.1(b) in section 4.1) and a *correct plan* w.r.t. the goal Fs (see proposition 4.1(c)). Thus, it easy to see that by the query 9 we formalize a special case of the planning problem, where the procedure definitions constrain the search space of reachable states in which to look for the wanted sequence.⁶

Example 2.7. Consider the domain description in example 2.6, with the difference that the robot also knows that the cover protecting *unit2*’s dial is not raised and that the dial is set on the position *off*. The query

$$\langle all_units_high \rangle (\mathcal{B}flow(unit1, high) \wedge \mathcal{B}flow(unit2, high))$$

amounts to ask whether it is possible to find a terminating execution of the procedure *all_units_high* (a plan) which leads to a state where all the air conditioning units of the room are blowing air with high speed. The plan is the following:

$$\begin{aligned} &turn_dial(unit1); go_to_unit(unit2); raise_cover(unit2); \\ &turn_dial(unit2); turn_dial(unit2). \end{aligned}$$

⁶ Note that, as a special case, we can define a procedure p which repeatedly selects any world action, so that all the world action sequences can be taken into account.

When the information available at planning time is incomplete and the procedure p includes sensing actions for acquiring the fresh information to be used for deciding what step to take next, the planning task as expressed above is not adequate. In particular, what action to perform next in a plan may depend on the outcome of a previous sensing action and such outcome can be known only at run time (when the agent actually consults its sensors). In this setting no linear sequence of actions can be demonstrated to achieve the goal. What we expect is that the planning process returns a *conditional plan* that achieves the goal no matter how the sensing turns out [33]. The plan has to be conditional because it must contain, for every possible outcome of the sensing, a legal course of actions that leads to the goal, in such a way that, when it will be executed and the sensors will actually return the missing information, it will be always possible to determine the next step to take toward the goal. Thus, in presence of sensing, by the query $\langle p \rangle Fs$ we will look for a conditional plan conforming to p which determines the actions to be executed for all possible results of the sensing actions. The resulted plan will have a tree structure where branches correspond to the possible outcome of sensing. Each execution of the plan can be proved to be a legal sequence of world actions which is guaranteed to lead to the goal state under certain assumptions on the outcome of the sensings (it follows from the property stated in 4.4, section 4.2). Intuitively, it ensures that the plan is *correct* w.r.t. the goal and that the plan is *executable*, i.e. when an agent executes it and consults the sensors, it can always determine the next step to take toward the goal. To deal with these matters, in section 4 we will define for our language a goal directed proof procedure, which, given a query $\langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle Fs$, extracts a linear plan to achieve the wanted goal, and a second one that in presence of sensing actions extracts conditional plans.

3. The persistency problem

The persistency problem is known in the literature on formalization of dynamic domains as the problem of specifying those fluents which remain unaffected by the execution of a given action. In our formalization, we provide a non-monotonic solution to the *persistency problem*. Intuitively, the problem is faced by using persistency assumptions: when an action is performed, any epistemic fluent F which holds in the state before executing the action is assumed to hold in the resulting state unless the action makes it false. As in [12], we model persistency assumptions by abductive assumptions: building upon the monotonic interpretation of a dynamic domain description we provide an abductive semantics to account for this non-monotonic behavior of the language.

3.1. The monotonic interpretation of a dynamic domain description

First of all, let us introduce some definitions. Given a dynamic domain description (Π, S_0) , let us call $\mathcal{L}_{(\Pi, S_0)}$ the propositional modal logic on which (Π, S_0) is based. The simple action clauses for primitive actions in $\Pi_{\mathcal{A}}$ and the initial beliefs in S_0 define a *theory* in $\mathcal{L}_{(\Pi, S_0)}$ that we denote with $\Sigma_{(\Pi, S_0)}$.

The axiomatization of $\mathcal{L}_{(\Pi, S_0)}$, called $\mathcal{S}_{(\Pi, S_0)}$, contains all the axioms and rules for propositional calculus, plus, for each modality the rule of *necessitation* and axiom schema K . Moreover, the axiomatization contains:

- for the operator \mathcal{B} , the axiom schema $D(\mathcal{B})$: $\mathcal{B}\varphi \supset \mathcal{M}\varphi$ (seriality);
- for the operator \square , the axiom schemas $T(\square)$: $\square\varphi \supset \square\square\varphi$ (reflexivity), $4(\square)$: $\square\varphi \supset \square\square\varphi$ (transitivity), and the interaction axiom $I(\square, a_i)$: $\square\varphi \supset [a_i]\varphi$, one for each world action $a_i \in \mathcal{A}$;

The operators “;” and “ \cup ” are ruled as usual in dynamic logic [30]. The test operator “?” is ruled by the axiom $\langle\psi?\rangle\varphi \equiv \psi \wedge \varphi$. Finally, the axiomatization includes the axioms $\Pi_{\mathcal{P}}$ and in $\Pi_{\mathcal{S}}$ characterizing complex actions and sensing actions, respectively.

The model theoretic semantics of the logic $\mathcal{L}_{(\Pi, S_0)}$ is given through a standard Kripke semantics with inclusion properties among the accessibility relations [2]. Let us define formally the notion of Kripke (Π, S_0) -interpretation for the logic $\mathcal{L}_{(\Pi, S_0)}$.

Definition 3.1 (Kripke semantics). A Kripke (Π, S_0) -interpretation M is a tuple $\langle W, \mathcal{R}_{\mathcal{B}}, \{\mathcal{R}_a : a \in \mathcal{A} \cup \mathcal{P} \cup \mathcal{S}\}, \mathcal{R}_{\square}, V \rangle$, where:

- W is a non-empty set of possible worlds;
- $\mathcal{R}_{\mathcal{B}}$ is a binary relation on W (the accessibility relation associated with \mathcal{B}) which is serial;
- every \mathcal{R}_a is a binary relation on W (the accessibility relation associated with $[a]$);
- \mathcal{R}_{\square} is a binary relations on W (the accessibility relation associated with \square) which is reflexive and transitive, and satisfies the condition $\mathcal{R}_{\square} \supseteq (\bigcup_{a_i} \mathcal{R}_{a_i})^*$, i.e. \mathcal{R}_{\square} contains the reflexive and transitive closure of the union of the \mathcal{R}_{a_i} (where the a_i ’s are atomic word actions in \mathcal{A}).
- V is a *valuation function*, that is a mapping from W and the set of fluent names to the set $\{\mathbf{T}, \mathbf{F}\}$.

Moreover, we define the accessibility relations for the complex actions built by means of operators “ \cup ”, “;” as usual in dynamic logic [30]:

- $\mathcal{R}_{\psi?} = \{(w, w) \mid M, w \models \psi\}$;
- $\mathcal{R}_{a;b} = \mathcal{R}_a \circ \mathcal{R}_b$, where “ \circ ” denotes the composition of binary relations;
- $\mathcal{R}_{a \cup b} = \mathcal{R}_a \cup \mathcal{R}_b$, where “ \cup ” denotes the union of binary relations.

Finally, we require that for each axiom $\langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle \varphi \supset \langle p_0 \rangle \varphi$ in $\Pi_{\mathcal{P}} \subseteq (\Pi, S_0)$, the following *inclusion property* on the accessibility relation holds:

$$\mathcal{R}_{p_0} \supseteq \mathcal{R}_{p_1} \circ \mathcal{R}_{p_2} \circ \dots \circ \mathcal{R}_{p_n}. \quad (10)$$

Note that, a p_i can be also a test. Similarly, we require that for each sensing axiom $[s]\varphi \equiv [\bigcup_{l \in \text{dom}(s)} s^{Bl}]\varphi$ in $\Pi_S \subseteq (\Pi, S_0)$, the following property on the accessibility relation holds:

$$\mathcal{R}_s \equiv \bigcup_{l \in \text{dom}(s)} \mathcal{R}_{s^{Bl}}. \quad (11)$$

The truth conditions are defined as usual. In particular:

- $M, w \models \mathcal{B}\varphi$, iff for all $w' \in W$ such that $(w, w') \in \mathcal{R}_B$, $M, w' \models \varphi$;
- $M, w \models \mathcal{M}\varphi$, iff there exists a $w' \in W$ such that $(w, w') \in \mathcal{R}_B$ and $M, w' \models \varphi$;
- $M, w \models [t]\varphi$, where t is either a world action a , or a sensing action s , or a procedure name p , or a test $\varphi?$, or a sequence $t; t'$, or a union $t \cup t'$, iff for all $w' \in W$ such that $(w, w') \in \mathcal{R}_t$, $M, w' \models \varphi$;
- $M, w \models \langle t \rangle \varphi$, where t is either a world action a , or a sensing action s , or a procedure name p , or a test $\varphi?$, or a sequence $t; t'$, or a union $t \cup t'$, iff there exists a $w' \in W$ such that $(w, w') \in \mathcal{R}_t$ and $M, w' \models \varphi$;
- $M, w \models \Box\varphi$ iff for all $w' \in W$ such that $(w, w') \in \mathcal{R}_\Box$, $M, w' \models \varphi$.

The set of all Kripke (Π, S_0) -interpretations is denoted by $\mathcal{M}_{(\Pi, S_0)}$. Given a Kripke (Π, S_0) -interpretation $M = \langle W, \mathcal{R}_B, \{\mathcal{R}_a: a \in \mathcal{A} \cup \mathcal{P} \cup \mathcal{S}\}, \mathcal{R}_\Box, V \rangle$ of $\mathcal{M}_{(\Pi, S_0)}$, we say that a formula φ of $\mathcal{L}_{(\Pi, S_0)}$ is *satisfiable in M* , if for some world $w \in W$ we have $M, w \models \varphi$. We say that φ is *valid in M* if for all worlds $w \in W$ $M, w \models \varphi$. Moreover, a formula φ is *satisfiable with respect to the class $\mathcal{M}_{(\Pi, S_0)}$* , if φ is satisfiable in some Kripke (Π, S_0) -interpretation in $\mathcal{M}_{(\Pi, S_0)}$ and *valid with respect to $\mathcal{M}_{(\Pi, S_0)}$* if it is valid in all Kripke (Π, S_0) -interpretations in $\mathcal{M}_{(\Pi, S_0)}$ (in this case, we write $\models \varphi$).

The axiom system $\mathcal{S}_{(\Pi, S_0)}$ is a *sound* and *complete* axiomatization with respect to $\mathcal{M}_{(\Pi, S_0)}$ [2,30].

3.2. The abductive semantics

The abductive semantics builds on monotonic logic $\mathcal{L}_{(\Pi, S_0)}$ and it is defined in the style of Eshghi and Kowalski's abductive semantics for negation as failure [25]. Let us denote by F an epistemic fluent literal, that is Bl or its negation $\neg Bl$, where l is a fluent literal. We define a new set of atomic propositions of the form $\mathbf{M}[a_1][a_2] \dots [a_m]F$ and we take them as being *abducibles*.⁷ Their meaning is that the epistemic fluent F can be assumed to hold in the state obtained by executing world actions a_1, a_2, \dots, a_m . Each abducible can be assumed to hold, provided it is consistent with the domain description (Π, S_0) and with other assumed abducibles. More precisely, we add to the axiom system of $\mathcal{L}_{(\Pi, S_0)}$ the *persistency axiom schema*:

$$[a_1][a_2] \dots [a_{m-1}]F \wedge \mathbf{M}[a_1][a_2] \dots [a_{m-1}][a_m]F \supset [a_1][a_2] \dots [a_{m-1}][a_m]F \quad (12)$$

⁷ Notice that here \mathbf{M} is not a modality. Rather, $\mathbf{M}\alpha$ is the notation used to denote a new atomic proposition associated with α . This notation has been adopted in analogy to default logic, where a justification $\mathbf{M}\alpha$ intuitively means “ α is consistent”.

where a_1, a_2, \dots, a_m ($m > 0$) are world actions, and F is an epistemic fluent. Its meaning is that, if F holds after the action sequence a_1, a_2, \dots, a_{m-1} , and F can be assumed to persist after action a_m (i.e., it is consistent to assume $\mathbf{M}[a_1][a_2] \dots [a_m]F$), then we can conclude that F holds after performing the sequence a_1, a_2, \dots, a_m .

Given a domain description (Π, S_0) , let \models be the satisfiability relation in the monotonic modal logic $\mathcal{L}_{(\Pi, S_0)}$ defined above. We denote with $\Sigma_{(\Pi, S_0)}$ the set of the simple action clauses for world actions in $\Pi_{\mathcal{A}}$ and the initial beliefs in S_0 .

Definition 3.2 (Abductive solution for a dynamic domain description). A set of *abducibles* Δ is an *abductive solution* for (Π, S_0) if, for every epistemic fluent F and for every world action sequence a_1, a_2, \dots, a_m :

- (a) $\forall \mathbf{M}[a_1][a_2] \dots [a_m]F \in \Delta, \Sigma_{(\Pi, S_0)} \cup \Delta \not\models [a_1][a_2] \dots [a_m]\neg F$,
- (b) $\forall \mathbf{M}[a_1][a_2] \dots [a_m]F \notin \Delta, \Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1][a_2] \dots [a_m]\neg F$.

Condition (a) is a *consistency* condition, which guarantees that each assumption cannot be assumed if its “complementary” formula holds. Condition (b) is a *maximality* condition which forces an abducible to be assumed, unless its “complement” is proved. When an action is applied in a certain state, persistency of those fluents which are not modified by the direct effects of the action, is obtained by maximizing persistency assumptions.

Let us now define the notion of abductive solution for a query in a domain description.

Definition 3.3 (Abductive solution for a query). Given a domain description (Π, S_0) and a query $\langle p_1; p_2; \dots; p_n \rangle Fs$, an *abductive solution for the query in* (Π, S_0) is defined to be an abductive solution Δ for (Π, S_0) such that $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle p_1; p_2; \dots; p_n \rangle Fs$.

The consistency of an abductive solution, according to definition 3.2, is guaranteed by the seriality of \mathcal{B} (from which $\neg(\mathcal{B}l \wedge \mathcal{B}\neg l)$ holds for every literal l). However the presence in Π of alternative action laws for an action a , which have mutually inconsistent effects and are applicable in the same state, may cause unintended solutions which are obtained by the contraposition of precondition laws. Let us consider the following example.

Example 3.1 (Unintended solution). Consider a domain description (Π, S_0) , where Π includes the following set of action and precondition laws for the world action a , and the initial epistemic state of the agent includes the epistemic fluents $\mathcal{B}q$, $\mathcal{B}\neg p$, and $\mathcal{B}r$ (see figure 3).

- (1) $\Box(\mathcal{B}r \supset \langle a \rangle \top)$,
- (2) $\Box(\mathcal{B}q \supset [a]\mathcal{B}p)$,
- (3) $\Box(\mathcal{B}p \supset [a]\mathcal{B}\neg p)$.

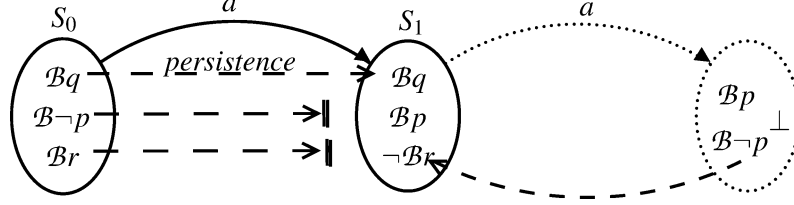


Figure 3. Unintended solution caused by contraposition from (1): $[a]\perp \supset \neg Br$.

Notice that the action laws (2) and (3), which rule the effects of action a , have *contradictory effects* Bp and $B\neg p$ (we recall that, by seriality, $B\neg p$ implies $\neg Bp$). The domain description has an unintended abductive solution Δ containing the assumption $\mathbf{M}[a]Bq$ but not $\mathbf{M}[a]Br$ because $\neg Br$ holds after the execution of action a . The solution is unintended as we would expect that the epistemic fluent Br , which holds in the initial state and is not modified by the direct effects of a , still holds by persistency after a is executed. Let us see why the persistency of Br is blocked after a (i.e., $\mathbf{M}[a]Br$ could not be assumed). After the execution of a , the epistemic fluent Bp holds as a direct effect; let us suppose that belief Bq persists ($\mathbf{M}[a]Bq \in \Delta$). By Bq and Bp in S_1 and the laws (2) and (3) we can infer that $[a]Bp$ and $[a]B\neg p$ hold in S_1 , that is $[a]\perp$ holds in S_1 (in other words, the formula $[a][a]\perp$ could be inferred from the theory (1)–(3) together with the assumption Δ). From this, by the contrapositive of the precondition law expressed in (1) – $[a]\perp \supset \neg Br$ – we can derive the formula $[a]\neg Br$, that is $\neg Br$ holds in S_1 . Deriving $[a]\neg Br$ blocks the persistency of Br from the initial state: $\mathbf{M}[a]Br$ cannot be assumed because of the condition (a) of definition 3.2 (consistency condition).

It is hard to accept that the persistency of an epistemic fluent (Br in the previous example) after the execution of a certain action is blocked even if there is no direct effect of that action that may affect the epistemic fluent. Indeed, intuitively, persistency of a fluent after performing an action must only depend on the effects of the action and on the truth values of the fluents in the state preceding the action execution.

Such unexpected solutions can be avoided by introducing an *e-consistency* requirement on domain descriptions, as for the language \mathcal{A} in [23]. Essentially, in order to avoid the use of the contraposition of a precondition law, we require that, for every set of action laws (for a given action) which may be applicable in the same state, the set of their effects is consistent.

Definition 3.4 (*e-consistency*). A domain description (Π, S_0) is *e-consistent* if for each world action $a \in \mathcal{A}$, for all the sets R

$$R = \{ \square(F_{s_1} \supset [a]F_1), \dots, \square(F_{s_n} \supset [a]F_n) \}$$

of a 's action laws in $\Pi_{\mathcal{A}} \subseteq \Pi$ s.t. the preconditions F_{s_1}, \dots, F_{s_n} are not mutually inconsistent, it holds that the set of effects $\{F_1, \dots, F_n\}$ is consistent.

Besides causing the existence of unintended solutions, the presence of actions with complementary effects in a domain description can also lead to have *multiple solutions* or *no solution*.

Example 3.2 (Multiple solutions and no solution). Consider example 3.1. If the rule (1) is substituted by the rule $\Box(\mathcal{B}q \supset \langle a \rangle \top)$ the resulting domain description has no solution. Instead, assume $\mathcal{B}t$ holds in the initial state and that that rule (1) is substituted by $\Box(\mathcal{B}r \wedge \mathcal{B}t \supset \langle a \rangle \top)$. The resulting domain description has two abductive solutions, the former containing $\mathbf{M}[a]\mathcal{B}q$ and $\mathbf{M}[a]\mathcal{B}r$, the latter containing $\mathbf{M}[a]\mathcal{B}q$ and $\mathbf{M}[a]\mathcal{B}t$.

Since world actions are deterministic w.r.t. the epistemic state (see proposition 4.3) and we do not have causal rules that can introduce non-deterministic effects or negative loops, assuming that the domain description is *e-consistent*, the following property holds for abductive solutions:

Proposition 3.1. Given an e-consistent dynamic domain description (Π, S_0) , there is a unique abductive solution for (Π, S_0) .

Existence and unicity of abductive solutions would not hold in a more general setting in which also causal rules are allowed (see, for instance, [29]). However, also in such a case, existence and unicity of abductive solutions can be enforced by putting suitable restrictions on the domain description.

Example 3.3. To see an example of abductive solution, let us consider the domain description (Π, S_0) , where Π is the tuple $(\Pi_{\mathcal{A}}, \Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$ of example 2.6 and S_0 includes the following epistemic fluent literals: $\mathcal{B}in_front_of(unit1)$, $\mathcal{B}\neg in_front_of(unit2)$, $\mathcal{B}\neg out_room$, $\mathcal{B}\neg cover_up(unit2)$, $\mathcal{B}cover_up(unit1)$. The simple action clauses in $\Pi_{\mathcal{A}}$ meet the e-consistency requirement. The query:

$$\langle go_to_unit(unit2) \rangle \langle raise_cover(unit2) \rangle (\mathcal{B}in_front_of(unit2) \wedge \mathcal{B}cover_up(unit2))$$

has an abductive solution Δ containing (among the others) the following abductive assumptions:

$$\begin{aligned} & \mathbf{M}[go_to_unit(unit2)]\mathcal{B}\neg out_room, \\ & \mathbf{M}[go_to_unit(unit2)]\mathcal{B}cover_up(unit1), \\ & \mathbf{M}[go_to_unit(unit2)]\mathcal{B}\neg cover_up(unit2), \\ & \mathbf{M}[go_to_unit(unit2)][raise_cover(unit2)]\mathcal{B}\neg out_room, \\ & \mathbf{M}[go_to_unit(unit2)][raise_cover(unit2)]\mathcal{B}cover_up(unit1), \\ & \mathbf{M}[go_to_unit(unit2)][raise_cover(unit2)]\mathcal{B}in_front_of(unit2), \\ & \mathbf{M}[go_to_unit(unit2)][raise_cover(unit2)]\mathcal{B}\neg in_front_of(unit1). \end{aligned}$$

In particular, since the protective cover of unit number 2 is not affected by the execution of the action $go_to_unit(unit2)$, the dial remains protected after moving (see the assumption $\mathbf{M}[go_to_unit(unit2)]\mathcal{B}\neg cover_up(unit2)$). Then, we can conclude that the effect $\mathcal{B}cover_up(unit2)$ holds after the action $raise_cover(unit2)$ is performed. Moreover, the assumption $\mathbf{M}[go_to_unit(unit2)] [raise_cover(unit2)]\mathcal{B}in_front_of(unit2)$ says that $\mathcal{B}in_front_of(unit2)$, which is made true by action $go_to_unit(unit2)$, persists after action $raise_cover(unit2)$.

4. Proof procedure: Finding correct plans

In section 4.1 we present a proof procedure which constructs a linear plan, by making assumptions on the possible result of sensing actions which are needed for the plan to reach the wanted goal. Such a proof procedure defines the computational part for the language presented in the previous sections. We call this logic programming language DyLOG.

In section 4.2 we introduce a proof procedure that constructs a conditional plan which achieves the goal for all the possible outcomes of the sensing actions.

4.1. Linear plan generation

In this section we introduce a *goal directed proof procedure* based on *negation as failure* (NAF) which allows a query to be proved from a given dynamic domain description. From a procedural point of view our non-monotonic way of dealing with the frame problem consists in using negation as failure, in order to verify that the *complement* of the epistemic fluent F is not made true in the state resulting from an action execution, while in the modal theory we adopted an abductive characterization to deal with persistency. However, it is well studied how to give an abductive semantics for NAF [25].

The first part of the proof procedure, denoted by “ \vdash_{ps} ” and presented in figure 4, deals with execution of complex actions, sensing actions, world actions and tests. The proof procedure reduces the complex and sensing actions in the query to a sequence of world actions and tests, and verifies if execution of the world actions is possible and if

$$\begin{array}{l}
 1) \frac{\bar{a}_{1\dots m} \vdash_{ps} \langle p'_1; \dots; p'_{n'}; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma}{\bar{a}_{1\dots m} \vdash_{ps} \langle p; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma} \quad \text{where } p \in \mathcal{P} \text{ and} \\
 \quad \langle p \rangle \varphi \subset \langle p'_1; \dots; p'_{n'} \rangle \varphi \in \Pi_{\mathcal{P}} \\
 2) \frac{\bar{a}_{1\dots m} \vdash_{fs} Fs' \quad \bar{a}_{1\dots m} \vdash_{ps} \langle \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma}{\bar{a}_{1\dots m} \vdash_{ps} \langle (Fs')?; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma} \\
 3) \frac{\bar{a}_{1\dots m} \vdash_{fs} Fs' \quad \bar{a}_{1\dots m}, a \vdash_{ps} \langle \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma}{\bar{a}_{1\dots m} \vdash_{ps} \langle a; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma} \quad \text{where } a \in \mathcal{A} \text{ and} \\
 \quad \square(Fs' \supset \langle a \rangle \top) \in \Pi_{\mathcal{A}} \\
 4) \frac{\bar{a}_{1\dots m} \vdash_{ps} \langle s^{Bl}; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma}{\bar{a}_{1\dots m} \vdash_{ps} \langle s; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \sigma} \quad \text{where } s \in \mathcal{S} \text{ and} \\
 \quad l \in dom(s) \\
 5) \frac{\bar{a}_{1\dots m} \vdash_{fs} Fs}{\bar{a}_{1\dots m} \vdash_{ps} \langle \varepsilon \rangle Fs \text{ w.a. } \sigma} \quad \text{where } \sigma = \bar{a}_{1\dots m}
 \end{array}$$

Figure 4. The derivation relation \vdash_{ps} . $\bar{a}_{1\dots m}$ is a_1, \dots, a_m and $\bar{p}_{2\dots n}$ is p_2, \dots, p_n .

the test actions are successful. To do this, it reasons about the execution of a sequence of world actions from the initial state and computes the values of fluents at different states. During a computation, a *state* is represented by a sequence of world actions a_1, \dots, a_m . The value of fluents at a state is not explicitly recorded but it is computed when needed in the computation. The second part of the procedure, denoted by “ \vdash_{fs} ” and presented in figure 5, allows the values of fluents in a state to be determined.

A query of the form $\langle p_1; p_2; \dots; p_n \rangle Fs$, where p_i , $1 \leq i \leq n$ ($n \geq 0$), is either a world action, or a sensing action, or a procedure name, or a test, succeeds if it is possible to execute p_1, p_2, \dots, p_n (in the order) starting from the current state, in such a way that Fs holds at the resulting state. In general, we will need to establish if a goal holds at a given state. Hence, we will write:

$$a_1, \dots, a_m \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs \text{ with answer (w.a.) } \sigma$$

to mean that the query $\langle p_1; p_2; \dots; p_n \rangle Fs$, i.e. $\langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle Fs$, can be proved from the domain description (Π, S_0) at the state a_1, \dots, a_m with answer σ , where σ is an action sequence $a_1, \dots, a_m, \dots, a_{m+k}$ which represents the state resulting by executing p_1, \dots, p_n in the current state a_1, \dots, a_m . We denote by ε the initial state.

The five rules of the derivation relation \vdash_{ps} in figure 4 define, respectively, *how to execute* procedure calls, tests, sensing actions and world actions. To execute a complex action p we non-deterministically replace the modality $\langle p \rangle$ with the modality in the antecedent of a suitable axiom for it (rule 1). To execute a test $(Fs)?$, the value of Fs is checked in the current state. If Fs holds in the current state, the test is simply eliminated, otherwise the computation fails (rule 2). To execute a world action a , first we need to verify if that action is possible by using the precondition laws. If these conditions hold we can move to a new state in which the action has been performed (rule 3). To execute a sensing action s (rule 4) we non-deterministically replace it with one of the world actions which define it (see section 2.2), that, when it is executable, will cause $\mathcal{B}l$ and $\mathcal{B}\neg l'$, for each $l' \in \text{dom}(s)$, with $l \neq l'$. Rule 5 deals with the case when there are no more actions to be executed. The sequence of world actions to be executed a_1, \dots, a_m has been already determined and, to check if Fs is true after a_1, \dots, a_m , proof rules 6–10 below are used.

The second part of the procedure (see figure 5) determines the derivability of an epistemic fluent conjunction Fs at a state a_1, \dots, a_m , denoted by $a_1, \dots, a_m \vdash_{fs} Fs$, and it is defined inductively on the structure of Fs . An epistemic fluent F holds at state a_1, a_2, \dots, a_m if: either F is an immediate effect of action a_m , whose preconditions hold in the previous state (rule 7a); or the last action, a_m , is an *ad hoc* primitive action s^F (introduced to model the sensing action s), whose effect is that of adding F to the state (rule 7b); or F holds in the previous state a_1, a_2, \dots, a_{m-1} and it persists after executing a_m (rule 7c); or a_1, a_2, \dots, a_m is the initial state and F holds in it. Notice that rule 7c allows to deal with the *frame problem*: F persists from a state a_1, a_2, \dots, a_{m-1} to the next state a_1, a_2, \dots, a_m unless a_m makes $\neg F$ true, i.e. it persists if $\neg F$ fails from a_1, a_2, \dots, a_m . In rule 7c **not** represents *negation as failure*. Moreover, rule 7a can deal with a more general form of simple action clauses than the ones presented in section 2.

$$\begin{array}{l}
6) \frac{}{\bar{a}_{1\dots m} \vdash_{fs} \top} \\
7a) \frac{\bar{a}_{1\dots m-1} \vdash_{fs} Fs'}{\bar{a}_{1\dots m} \vdash_{fs} F} \quad \text{where } m > 0 \text{ and} \\
\quad \quad \quad \square(Fs' \supset [a_m]F) \in \Pi_{\mathcal{A}} \\
7b) \frac{}{\bar{a}_{1\dots m} \vdash_{fs} F} \quad \text{if } a_m = s^F \\
7c) \frac{\text{not } \bar{a}_{1\dots m} \vdash_{fs} \neg F \quad \bar{a}_{1\dots m-1} \vdash_{fs} F}{\bar{a}_{1\dots m} \vdash_{fs} F} \quad \text{where } m > 0 \\
7d) \frac{}{\varepsilon \vdash_{fs} F} \quad \text{if } F \in S_0 \\
8) \frac{\bar{a}_{1\dots m} \vdash_{fs} Fs_1 \quad \bar{a}_{1\dots m} \vdash_{fs} Fs_2}{\bar{a}_{1\dots m} \vdash_{fs} Fs_1 \wedge Fs_2} \\
9) \frac{\bar{a}_{1\dots m} \vdash_{fs} \mathcal{B}l}{\bar{a}_{1\dots m} \vdash_{fs} \neg \mathcal{B}\neg l}
\end{array}$$

Figure 5. The derivation relation \vdash_{fs} . $\bar{a}_{1\dots m}$ is a_1, \dots, a_m and $\bar{p}_{2\dots n}$ is p_2, \dots, p_n .

In particular, it deals with action laws of the form $\square(Fs \supset [a]F)$ and precondition laws of the form $\square(Fs \supset \langle a \rangle \top)$, where Fs is an arbitrary conjunction of epistemic fluents and F is an epistemic fluent, respectively. Rule 8 deals with the conjunction while rule 9 allows $\neg \mathcal{B}\neg l$ to be concluded from $\mathcal{B}l$ which is justified by the property of seriality of the belief modality.⁸

A *proof* for a query of the form $\langle p_1; p_2; \dots; p_n \rangle Fs$ from a dynamic domain description (Π, S_0) at state a_1, \dots, a_m with answer σ is a finite tree constructed using rules 1–9 described above, such that:

- the root is labelled with $a_1, \dots, a_m \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ w.a. σ ;
- the leaves have the form either $a_1, \dots, a_m \vdash_{fs} \top$, or $a_1, \dots, a_m \vdash_{fs} F$ where a_m is s^F , or $\varepsilon \vdash_{fs} F$ where $F \in S_0$.

σ is an action sequence $a_1, \dots, a_m, \dots, a_{m+k}$ which represents the state resulting by executing p_1, \dots, p_n in the current state a_1, \dots, a_m . We say that a query $\langle p_1; p_2; \dots; p_n \rangle Fs$ *succeeds* from a dynamic domain description (Π, S_0) with answer σ if it has a *proof* in the initial state ε with the *execution trace* σ as answer, that is $\varepsilon \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ . Notice that the proof procedure does not perform any consistency check on the computed abductive solution. However, under the assumption that the domain description is e-consistent and that the beliefs on the initial state S_0 are consistent, soundness of the proof procedure above can be proved w.r.t. any acceptable solution.

Theorem 4.1 (Soundness). Let (Π, S_0) be an e-consistent dynamic domain description and let $\langle p_1; p_2; \dots; p_n \rangle Fs$ be a query. For every abductive solution Δ for (Π, S_0) , for every answer σ , if $\langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds from (Π, S_0) with answer σ , then $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle p_1; p_2; \dots; p_n \rangle Fs$.

⁸ The proof procedure works assuming that $\mathcal{M}l$ is replaced by $\neg \mathcal{B}\neg l$.

The proof (in appendix A) is by induction on the rank of the derivation of the query, and it makes use of a soundness and completeness result for the monotonic part of the proof procedure presented in this section w.r.t. the monotonic part of the semantics. Indeed, if the assumptions $\mathbf{M}[a_1][a_2] \dots [a_m]F$ are regarded as facts rather than abducibles and they are added to the program, the non-monotonic step 7c in the proof procedure can be replaced by a monotonic one. The resulting monotonic proof procedure can be shown to be sound and complete with respect to the Kripke semantics of the modal logic $\mathcal{L}_{(\Pi, S_0)}$.

Our proof procedure computes just one solution, while abductive semantics may give multiple solutions for a domain description. However, as we already mentioned, under the condition that a domain description is *e-consistent* there is a unique abductive solution for the domain description (see proposition 3.1). Under such restriction, we argue that our proof procedure is also *complete*.

Since a query $\langle p_1; p_2; \dots; p_n \rangle Fs$ is an existential formula, a successful answer σ represents a possible execution of the sequence p_1, p_2, \dots, p_n . Indeed, for the answer σ we can prove the proposition 4.1. Property (a) says that σ is a *possible execution* of p_1, p_2, \dots, p_n , (b) says that σ is a *legal sequence* of actions, and, finally, (c) says that the plan σ is *correct* w.r.t. Fs .

Proposition 4.1. Let (Π, S_0) be an e-consistent dynamic domain description and let $\langle p_1; p_2; \dots; p_n \rangle Fs$ be a query. For every abductive solution Δ for (Π, S_0) , for every answer σ , if $\varepsilon \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ then:

- (a) $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle \sigma \rangle Fs \supset \langle p_1; p_2; \dots; p_n \rangle Fs$,
- (b) $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle \sigma \rangle \top$,
- (c) $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\sigma] Fs$.

Let us remember that σ is a sequence of world actions of the domain. Then, before proving the above proposition about σ , we prove some useful property concerning world actions of our domain descriptions.

The following proposition states that world actions in our domain descriptions are deterministic w.r.t. the epistemic state, i.e. there is only one epistemic state reachable by executing a world action a in a given epistemic state.

Proposition 4.2. Let (Π, S_0) be an e-consistent dynamic domain description and let G be a query of the form $\langle p_1; p_2; \dots; p_n \rangle Fs$. For every abductive solution Δ for (Π, S_0) , the following property holds:

$$\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a \rangle G \supset [a]G$$

where a is a world action of (Π, S_0) .

Proof. In order to prove this property we make use of the soundness and completeness results of the monotonic proof procedure \vdash_{Δ} w.r.t. the Kripke semantics. Let us assume

$\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a \rangle G$ and we prove $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a]G$. Since $\langle a \rangle G$ has the form of a query, by the completeness result of theorem A.3, our hypothesis implies that $\langle a \rangle G$ succeeds from (Π, S_0) , i.e. $\varepsilon \vdash_{\Delta} \langle a \rangle G$. But if $\varepsilon \vdash_{\Delta} \langle a \rangle G$, then by definition of \vdash_{Δ} (rule 3 of theorem A.1) there exists a proof of Fs from (Π, S_0) in the state a , i.e. $a \vdash_{\Delta} G$. Then, by the soundness result of theorem A.1, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a]G$. \square

Proposition 4.2 can be easily generalized to world action sequences.

Proposition 4.3. Let (Π, S_0) be an e-consistent dynamic domain description and let G be a query of the form $\langle p_1; p_2; \dots; p_n \rangle Fs$. For every abductive solution Δ for (Π, S_0) , the following property hold:

$$\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a_1; a_2; \dots; a_m \rangle G \supset [a_1; a_2; \dots; a_m]G$$

where where a_1, a_2, \dots, a_m ($m > 0$) are world actions of (Π, S_0) .

Now we are in the position to give the proof of the proposition 4.1.

Proof of proposition 4.1. Property (a). Our hypothesis is $\varepsilon \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ . Let us assume $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle \sigma \rangle Fs$. It means that for each (Π, S_0) -interpretation and world w , $M, w \models \Sigma_{(\Pi, S_0)} \cup \Delta$ implies $M, w \models \langle \sigma \rangle Fs$. $M, w \models \langle \sigma \rangle Fs$ iff it exists a world w' s.t. $(w, w') \in \mathcal{R}_{\sigma}$, where σ is an action sequence a_1, \dots, a_m . Then, since by hypothesis the action sequence σ is an answer for the query $\langle p_1; p_2; \dots; p_n \rangle Fs$, it is easy to prove that $(w, w') \in \mathcal{R}_{p_1; \dots; p_n}$ too. It can be done by analyzing that part of the derivation of $\langle p_1; p_2; \dots; p_n \rangle Fs$ which deals with the reductions of the complex actions in the query to the action sequence a_1, \dots, a_m . In particular, the path connecting w and w' labelled by $\mathcal{R}_{p_1; \dots; p_n}$ is constructed by collecting: (i) the inclusion relations expressed by the procedure axioms, when rule 1 has been applied for reducing a procedure definition; (ii) the inclusion relations expressed by the suitable sensing axioms, when rule 4 has been applied for reducing a sensing action; (iii) the accessibility relations expressed by the suitable test axioms, when rule 2 has been applied for reducing a test.

Property (b). Our hypothesis is $\varepsilon \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ , i.e. it exists a proof Υ for the query $\langle p_1; p_2; \dots; p_n \rangle Fs$ from (Π, S_0) at state ε w.a. σ . Then, by definition of \vdash_{ps} , we can deduce that during the proof complex actions in the query have been reduced to the sequence of world actions contained in the answer σ , i.e. there is a sub-proof of Υ with root $a_1; \dots; a_m \vdash_{ps} \langle \varepsilon \rangle Fs$ w.a. σ , where $\sigma = a_1; \dots; a_m$. From it, by the soundness result of theorem A.4, it follows $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a_1; a_2; \dots; a_m \rangle Fs$, and, then, $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a_1; a_2; \dots; a_m \rangle \top$.

Property (c). Let us consider the proof of (b) given above. From our hypothesis $\varepsilon \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ we obtain $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle a_1; a_2; \dots; a_m \rangle Fs$, then, using proposition 4.3, by modus ponens, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; a_2; \dots; a_m]Fs$, and, since $a_1; \dots; a_m = \sigma$, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\sigma]Fs$. \square

4.2. Conditional plan generation

In this section we introduce a proof procedure that constructs a conditional plan which achieves the goal for all the possible outcomes of the sensing actions. Let us start with an example.

Example 4.1. Consider example 2.6 and the query

$$\langle all_units_high \rangle (\mathcal{B}flow(unit1, high) \wedge \mathcal{B}flow(unit2, high)).$$

We want to find an execution of *all_units_high* reaching a state where all the air conditioning units of the room are blowing air with high speed. When in the initial state the agent does not have information about the *unit2* (i.e. it is unknown if the cover protecting the dial of *unit2* is raised and which is the position of the unit dial) the action sequence the agent has to perform to achieve the goal depends on the outcome of the sensing actions *sense_cover(unit2)* and *sense_dial(unit2)*. Indeed, after performing the action sequence *turn_dial(unit1); go_to_unit(unit2)* the robot has to execute a sensing on *unit2* cover in order to know if it is raised or not. The result of sensing determines the robot future course of actions: if it comes to know that the cover it is raised, it can execute directly a sensing on the dial for knowing if is in the position off, low, or high, otherwise it has to raise the dial cover before to proceed with the sensing. Going on, the result of *sense_dial(unit2)* determines the further robot behavior: if it comes to know that the unit flow is off, it has to turn the dial twice in order to reach the position *high*, if it discover that the unit flow is low, it has to turn the dial only once in order to reach the desired position, finally, if it comes to know that the flow is already high it does not perform any action.

Given the query above, the proof procedure described in the previous section extracts, among the others, the following world action sequences, making assumptions on the possible results of *sense_cover(unit2)* and *sense_dial(unit2)*:

- *turn_dial(unit1); go_to_unit(unit2); sense_cover(unit2)* ^{$\mathcal{B}cover_up(unit2)$} ;
sense_dial(unit2) ^{$\mathcal{B}flow(unit2, low)$} ; *turn_dial(unit2)*,
- *turn_dial(unit1); go_to_unit(unit2); sense_cover(unit2)* ^{$\mathcal{B}\neg cover_up(unit2)$} ;
raise_cover(unit2); sense_dial(unit2) ^{$\mathcal{B}flow(unit2, high)$} .

Intuitively, in the first solution the procedure assumed that the sensing actions *sense_cover(unit2)* and *sense_dial(unit2)* cause to belief that the protecting cover is up and the unit flow is low, respectively. Under these assumptions, it plans to execute the *turn_dial(unit2)* action. Instead, in the second solution the procedure assumed that after sensing the dial results to be protected by the cover and then it plans to raise the cover before sensing the dial. Since after sensing the dial the procedure assumed that the unit flow is already high, it does not plan any further action.

Instead the proof procedure we are going to present, given the same query, will look for a conditional plan that achieves the goal $(\mathcal{B}\neg\text{open}(\text{unit1}) \wedge \mathcal{B}\neg\text{open}(\text{unit2}))$ for any outcome of the sensing actions, as follows:

```

turn_dial(unit1);
go_to_unit(unit2);
sense_cover(unit2);
  (( $\mathcal{B}\text{cover\_up}(\text{unit2})?$ );
    sense_dial(unit2);
      (( $\mathcal{B}\text{flow}(\text{unit2}, \text{high})?$ );  $\cup$ 
        ( $\mathcal{B}\text{flow}(\text{unit2}, \text{low})?$ );
          turn_dial(unit2);  $\cup$ 
            ( $\mathcal{B}\text{flow}(\text{unit2}, \text{off})?$ );
              turn_dial(unit2); turn_dial(unit2));  $\cup$ 
        ( $\mathcal{B}\neg\text{cover\_up}(\text{unit2})?$ );
          raise_cover(unit2); sense_dial(unit2);
            (( $\mathcal{B}\text{flow}(\text{unit2}, \text{high})?$ );  $\cup$ 
              ( $\mathcal{B}\text{flow}(\text{unit2}, \text{low})?$ );
                turn_dial(unit2);  $\cup$ 
                  ( $\mathcal{B}\text{flow}(\text{unit2}, \text{off})?$ );
                    turn_dial(unit2); turn_dial(unit2)))

```

Intuitively, given a query $\langle p \rangle Fs$, the proof procedure we are going to define computes a conditional plan σ (if there is one), which determines the actions to be executed for all possible results of the sensing actions. All the executions of the conditional plan σ are possible behaviors of the procedure p .

Definition 4.1 (Conditional plan). A conditional plan is defined inductively by the followings:

1. a (possibly empty) sequence of world actions $a_1; a_2; \dots; a_n$ is a conditional plan;
2. if $a_1; a_2; \dots; a_n$ is a world action sequence, $s \in \mathcal{S}$ is a sensing action, and $\sigma_1, \dots, \sigma_t$ are conditional plans then $a_1; a_2; \dots; a_n; s; ((\mathcal{B}l_1?); \sigma_1 \cup \dots \cup (\mathcal{B}l_t?); \sigma_t)$ is a conditional plan, where $l_1, \dots, l_t \in \text{dom}(s)$.

Given a query $\langle p_1; p_2; \dots; p_n \rangle Fs$ the proof procedure constructs, as answer, a conditional plan σ such that:

1. all the executions of σ are possible executions of $p_1; p_2; \dots; p_n$, and
2. all the executions of σ lead to a state in which Fs holds.

The proof procedure is defined on the basis of the previous relations \vdash_{ps} and \vdash_{fs} . We simply need to replace relation \vdash_{ps} with the relation $\vdash_{ps_{cond}}$ that has all the rules of \vdash_{ps} but rule 4 (dealing with the execution of sensing actions). rule 4 in $\vdash_{ps_{cond}}$ is replaced by the following rule 4-bis:

$$4\text{-bis}) \frac{\forall l_i \in \mathcal{F}, \bar{a}_{1\dots m} \vdash_{ps_{cond}} \langle s^{Bl_i}; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \bar{a}_{1\dots m}; s^{Bl_i}; \sigma'_i}{\bar{a}_{1\dots m} \vdash_{ps_{cond}} \langle s; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \bar{a}_{1\dots m}; s; ((Bl_1?); \sigma'_1 \cup \dots \cup (Bl_t?); \sigma'_t)}$$

where $s \in \mathcal{S}$ and $\mathcal{F} = \{l_1, \dots, l_t\} = \text{dom}(s)$, $\bar{a}_{1\dots m}$ is a_1, \dots, a_m , and $\bar{p}_{2\dots n}$ is p_2, \dots, p_n . As a difference with the previous proof procedure, when a sensing action is executed, the procedure has to consider all possible outcomes of the action, so that the computation splits in multiple branches. If all branches lead to success, it means that the main query succeeds for all the possible results of action s . In such a case, the conditional plan σ will contain the σ'_i 's as alternative sub-plans and every branch of σ is actually a linear plan.

Definition 4.2. Let σ be a conditional plan as defined in section 4.2 and σ' a linear plan as defined in section 4.1. We say that $\sigma' \subseteq \sigma$ if and only if:

- σ is equal to σ' , or
- σ is $a_1; \dots; a_n; s; ((Bl_1?); \sigma_1 \cup \dots \cup (Bl_t?); \sigma_t)$, where $s \in \mathcal{S}$, and $l_1, \dots, l_t \in \text{dom}(s)$, and σ' is $a_1; \dots; a_n; s^{Bl_i}; \sigma''$, where $\sigma'' \subseteq \sigma_i$, for some $i \in \{1, \dots, t\}$.

The intuition is that the linear plan σ' is included in the conditional plan σ .

Proposition 4.4. Let (Π, S_0) be a dynamic domain description, G a query and a_1, \dots, a_m a state. Then, if $a_1, \dots, a_m \vdash_{ps_{cond}} G$ with answer the conditional plan σ , then $a_1, \dots, a_m \vdash_{ps} G$ with answer σ' , for any $\sigma' \subseteq \sigma$.

Proof. The proof is by induction of the height of the $\vdash_{ps_{cond}}$ -proof Υ for the query G . If the height h of Υ is 1, the Υ is an axiom and this lemma holds trivially.

By inductive hypothesis the lemma holds for queries whose proof Υ has height less than or equal to h . Let us prove it for $h + 1$. There are a case for each rule in which Υ can terminate. All cases but rule 4-bis are an easy application of inductive hypothesis. Let us consider the case of rule 4-bis.

$$\frac{\forall l_i \in \mathcal{F}, \bar{a}_{1\dots m} \vdash_{ps_{cond}} \langle s^{Bl_i}; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \bar{a}_{1\dots m}; s^{Bl_i}; \sigma_i}{\bar{a}_{1\dots m} \vdash_{ps_{cond}} \langle s; \bar{p}_{2\dots n} \rangle Fs \text{ w.a. } \bar{a}_{1\dots m}; s; ((Bl_1?); \sigma_1 \cup \dots \cup (Bl_t?); \sigma_t)}$$

where $s \in \mathcal{S}$ and $\mathcal{F} = \{l_1, \dots, l_t\} = \text{dom}(s)$, $\bar{a}_{1\dots m}$ is a_1, \dots, a_m , and $\bar{p}_{2\dots n}$ is p_2, \dots, p_n . Now, by induction hypothesis, we have that $a_1, \dots, a_m \vdash_{ps} \langle s^{Bl_i}; p_2; \dots; p_n \rangle Fs$ with answer $a_1; \dots; a_m; s^{Bl_i}; \sigma'_i$, for any $\sigma'_i \subseteq \sigma_i$, for all $l_i \in \mathcal{F}$. Therefore, it easy to see that, by applying rule 4, $a_1, \dots, a_m \vdash_{ps} \langle s; p_2; \dots; p_n \rangle Fs$ with answer $a_1; \dots; a_m; s^{Bl_i}; \sigma'_i$, for any $\sigma'_i \subseteq \sigma_i$, for all $l_i \in \mathcal{F}$, that is the thesis. \square

As a corollary of proposition 4.4 we have that proposition 4.1 holds for every branch of a condition plan. The following theorem states the soundness of the proof procedure for generating conditional plans.

Theorem 4.2 (Soundness). Let (Π, S_0) be a dynamic domain description and let $\langle p_1; p_2; \dots; p_n \rangle Fs$ be a query. For every abductive solution Δ for (Π, S_0) , for every answer σ , if $\langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds from (Π, S_0) with answer σ , then $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle p_1; p_2; \dots; p_n \rangle Fs$.

Proof. If $a_1, \dots, a_m \vdash_{ps_{cond}} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer the conditional plan σ then, by proposition 4.4, $a_1, \dots, a_m \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ' , where $\sigma' \subseteq \sigma$. Therefore, by theorem A.4, we can conclude the thesis. \square

Finally, by the following proposition, we state that an extracted conditional plan σ is correct w.r.t. the conjunction of epistemic fluents Fs and the initial situation S_0 . In particular, this means that executing the plan σ (constructed by the procedure) always leads to a state in which Fs holds, for all the possible results of the sensing actions.

Proposition 4.5. Let (Π, S_0) be a dynamic domain description and let $\langle p_1; p_2; \dots; p_n \rangle Fs$ be a query. For every abductive solution Δ for (Π, S_0) , for every answer σ , if $\langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds from (Π, S_0) with answer σ , then $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\sigma]Fs$.

In order to prove this property, we need before to prove the following lemma.

Lemma 4.1. Let (Π, S_0) be a dynamic domain description, G a query and a_1, \dots, a_m a state. Then, if $a_1, \dots, a_m \vdash_{ps_{cond}} G$ with answer the conditional plan σ , then $\Sigma_{(\Pi, S_0)} \models [a_1; \dots; a_m][\bigcup_{\sigma' \subseteq \sigma} \sigma']G \supset [a_1; \dots; a_n][\sigma]G$.

Proof. The proof is by induction on the structure of the conditional plan σ . The case σ is $a_1; \dots; a_n$ is trivial. Let us suppose that σ is $a_1; \dots; a_m; s; ((\mathcal{B}_1?); \sigma_1 \cup \dots \cup (\mathcal{B}_t?); \sigma_t)$, and, by inductive hypothesis this lemma holds for the conditional plans σ_i , $1 \leq i \leq t$. Now, by hypothesis, $\models \Sigma_{(\Pi, S_0)}$ and $\models [a_1; \dots; a_m][\bigcup_{\sigma' \subseteq \sigma} \sigma']G$, that is $\models [a_1; \dots; a_n; s^{\mathcal{B}_i}; \sigma_i'']G$, for all $\sigma_i'' \subseteq \sigma_i$, where $1 \leq i \leq t$. Then, for each interpretation M , for each world w , $M, w \models [a_1; \dots; a_n; s^{\mathcal{B}_i}; \sigma_i'']G$, for all $\sigma_i'' \subseteq \sigma_i$, where $1 \leq i \leq t$. Therefore, by definition of satisfiability, we have $M, w' \models [s^{\mathcal{B}_i}; \sigma_i'']G$, for any world w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_n}$, and $M, w'' \models [\sigma_i'']G$, for any world w'' s.t. $(w', w'') \in \mathcal{R}_{s^{\mathcal{B}_i}}$, for all $\sigma_i'' \subseteq \sigma_i$, where $1 \leq i \leq t$. By inductive hypothesis we have that $M, w'' \models [\sigma_i]G$, where $1 \leq i \leq t$. Since (w', w'') belongs $\mathcal{R}_{s^{\mathcal{B}_i}}$, $\square(\top \supset [s^{\mathcal{B}_i}]\mathcal{B}_i) \in \Pi$ and, by hypothesis, $\models \Sigma_{(\Pi, S_0)}$ we have that $M, w' \models \top \supset [s^{\mathcal{B}_i}]\mathcal{B}_i$, that is $M, w' \models \mathcal{B}_i$. Then, it follows that $(w'', w'') \in \mathcal{R}_{s^{\mathcal{B}_i}}$ and, therefore, $M, w'' \models [(\mathcal{B}_i?); \sigma_i]G$. Now, since $\mathcal{R}_s \supseteq \bigcup_{l \in \text{dom}(s)} \mathcal{R}_{s^{\mathcal{B}_l}}$ and, therefore, (w', w'') belongs to \mathcal{R}_s too, we have that $M, w' \models [s; (\mathcal{B}_i?); \sigma_i]G$, for all i , $1 \leq i \leq t$. That is, $M, w' \models [s; ((\mathcal{B}_1?); \sigma_1 \cup \dots \cup (\mathcal{B}_t?); \sigma_t)]G$. Since this holds for any w' s.t. $(w, w') \in$

$\mathcal{R}_{a_1; \dots; a_n}$, it follows that $M, w \models [a_1; \dots; a_n; s; ((\mathcal{B}l_1?); \sigma_1 \cup \dots \cup (\mathcal{B}l_t?); \sigma_t)]G$, that is the thesis. \square

Proof of proposition 4.5. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [\sigma]Fs$. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [\sigma]Fs$. By hypothesis, we have that $\varepsilon \vdash_{ps_{cond}} \langle p_1; \dots; p_n \rangle Fs$ with answer σ , then, by proposition 4.4, we have that $\forall \sigma' \subseteq \sigma$, $\varepsilon \vdash_{ps} \langle p_1; \dots; p_n \rangle Fs$ with answer σ' . By proposition 4.1, $\forall \sigma' \subseteq \sigma$, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\sigma']Fs$, that is $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\bigcup_{\sigma' \subseteq \sigma} \sigma']Fs$, and, by lemma 4.1, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [\sigma]Fs$. \square

5. Implementation and web applications

In this section, first we will briefly sketch some issues that arised in implementing the DyLOG language, then we will describe our experience in using DyLOG as an agent logic programming language to implement Adaptive Web Applications.

5.1. Implementation

A DyLOG interpreter has been implemented in Sicstus Prolog. This implementation allows DyLOG to be used as an ordinary programming language for *executing* procedures which specify the behavior of an agent, but also for *reasoning* about them, by extracting linear or conditional plans. The plan extraction process of the interpreter is straightforward implementation of the proof procedure contained in the theoretical specification of the language.

For sake of readability, in the DyLOG implementation we adopted an English-like notation for the modal formulas of a domain description, that is action laws for world actions (AL), precondition laws for world actions (PL) sensing definitions for sensing action (SD), procedure definitions for complex actions (PD), beliefs about the initial situation (IB). Then our interpreter expects a program having the usual components with the following notation:

$$\begin{array}{ll}
 \text{AL:} & \Box(Fs \supset [a]F) \quad \rightsquigarrow a \text{ causes } F \text{ if } Fs, \\
 \text{PL:} & \Box(Fs \supset \langle a \rangle true) \quad \rightsquigarrow a \text{ possible if } Fs, \\
 \text{SD:} & [s]\varphi \equiv [s^{\mathcal{B}l} \cup s^{\mathcal{B}^{-l}}]\varphi \quad \rightsquigarrow s \text{ senses } l, \\
 \text{PD:} & \langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \dots \langle p_n \rangle \varphi \rightsquigarrow p_0 \text{ is } p_1, \dots, p_n', \\
 \text{IB:} & F \quad \rightsquigarrow \text{obs } (F).
 \end{array}$$

Notice that in our implementation we do not explicitly use the epistemic operator \mathcal{B} : if a fluent f (or its negation $\neg f$) is present in a state, it is intended to be believed, unknown otherwise. Thus each fluent can have one of the three values: true, false or unknown. In general, in order to test the value of a fluent f in clauses of the kind (PD) we use the notation “ $?(f)$ ”; the notation $?(u(f))$ is used to test if a fluent is unknown (i.e. to test if neither f nor $\neg f$ is present in the state). Let us mention that the implementation deals with domain descriptions containing a simple form of *causal laws* and *functional fluents* with associated finite domain, which are not explicitly treated in this paper.

5.1.1. On-line and hypothetical execution

DyLOG programs are executed by an interpreter which is a straightforward implementation of the proof procedure in section 4. Our interpreter allows to interleave two kinds of execution of programs, that we will call on-line and hypothetical execution respectively.

On the one hand, the procedures that specify the agent behavior can be executed on-line as in standard programming: the agent chooses an action among the ones that are legally executable at a certain point of the program and it must commit to it, thus there is no possibility of undoing it. In fact when the interpreter executes an action in this modality it is *not allowed to backtrack* by retracting the effects of the action. In general, the on-line execution of a world action modifies the agent state according to the action and causal laws, but when during the execution we encounter a sensing action, we must wait to know the outcome of the sensing (an external input) before to update the agent state and proceed. Moreover, on-line execution of an action can have a real effect when the agent application is set in a real environment, such as for instance moving a robot or sending a message. This can be specified in DyLOG by associating with each primitive action some Prolog code that implements the effects of the action on the world (the association is done by means of the keyword **performs**). In a setting where the DyLOG program is designed for controlling a robot such code could contain the call to a robot controller, while in a web setting, it could contain instructions for requesting to the actual execution device to send a given web page to the browser (see section 5.2).

On the other hand, a rational agent must be able to cope with complex or unexpected situations by *reasoning about the effects* of a procedure before executing it. Thus before to commit to a given behavior the interpreter can be asked to hypothetically execute the procedure. In this modality the agent will *reason* on possible sequences of actions by exploring different alternatives and extract a plan that achieves the desired goal. In order to deal with this hypothetical execution, the DyLOG implementation provides a metapredicate $plan(Fs \text{ after } p, as)$, where p is a procedure, Fs a condition on the goal state and as a sequence of primitive actions. The procedure p can be non-deterministic. When p does not contain sensing actions, the predicate $plan$ will extract from it a sequence as of primitive actions, a plan, corresponding to a possible execution of the procedure, leading to a state in which Fs holds, starting from the current state. Such predicate implements the query of form $\langle p \rangle Fs$ considered in the previous section, which ask for a terminating execution of p leading to a state in which Fs holds.⁹ Thus, it works by executing p in the same way as the on-line interpreter of the language, with a main differences: primitive actions are executed without any effect on the external environment, and, as a consequence, they are backtrackable. When the planning predicate $plan(Fs \text{ after } p, as)$ is applied to a procedure p that contains sensing actions, the interpreter simply implements the proof procedure in section 4.2: it looks ahead over sensing actions and tries to extract a *conditional plan*, that is guaranteed to lead to a state where

⁹ Notice that the predicate $plan$ can be also applied to action sequences: $plan((Fs \text{ after } a_1, \dots, a_n), _)$. In this case it implements the *temporal projection task*.

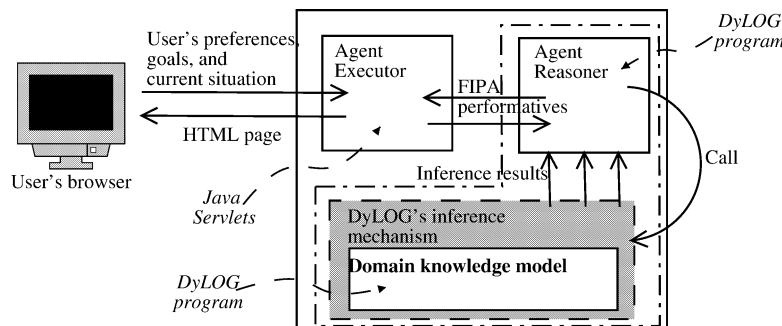


Figure 6. A sketch of the WLOG multiagent system.

F_s holds no matter the sensing turns out (branches correspond to the possible outcomes of sensing).

Further details are here omitted. Both the interpreter and the programming manual are available at [1].

5.2. Web applications

In the last years the use of DyLOG as specification language for implementing intelligent software agents have been experimented in significant agent based scenarios. In particular, our running application project fits in the area of intelligent adaptive systems accessible via web [4,8]. Most of the work carried on in this field [15] concerns the personalization of the site language and graphics and sometimes also of its contents, e.g., in the case of news portals the system will select the information to show according to the specific interests of the users. The most successful approach is based on user models, which are prototypes of users, sometimes refined after the interaction of the users with the site. However, there are applications, such as recommendation systems, which are designed to help users to solve problems of interest, where the user goals may vary at every connection, therefore cannot be inferred from the past user's behavior or from his/her general model. In such contexts the possibility to reason about actions and to adapt the system behavior to the current user goals can play a very important role in order to achieve a more complete adaptation.

In this framework in a set of papers [4,8–10] we proposed an approach based on logic agents, where adaptation is based on the reasoning capabilities of a DyLOG rational agent, applied to a declarative description of the domain (internal to the system) and meant to achieve the goals of the specific user. Recently we focused on an *adaptive tutoring* scenario and developed a web-based multi-agent system, called Wlog, that supports students in the process of constructing and validating a study plan by adapting to the learning goal and to the declared competence of the student.

5.2.1. A tutoring system

The system WLog has a *multi-agent* architecture, sketched in figure 6. The server-side of the system consists mainly of two kinds of agents: *reasoners* and *executors*. The

kernel of the system are the reasoners that have been implemented as DyLOG agents. The executors are Java servlets embedded in a Tomcat web server, playing the role of an interface among the rational agents and the users; they mainly produce HTML pages according to the directives sent by the DyLOG agents or, when necessary, forward data to the DyLOG agents themselves.

Let us focus on the implementation of the reasoners as DyLOG agents. Our reasoners work on a dynamic domain description, where the basic actions that can be executed are of the kind “attend course X ”. Effects and preconditions of actions (courses) are expressed by simple action clauses and are essentially given in terms of a set of abstract *competences*, possibly connected by causal relationships. The set of all the possible competences and of their relations defines an *ontology*. *Complex professional expertise* are described composing courses in predefined schemas by means of DyLOG procedure definitions. Such procedures are used by the agent to accomplish the task to build a study plan, i.e. a sequence of courses for achieving a certain high-level competence. Working at the level of competences is close to human intuition and enables the application of both goal-directed reasoning processes and explanation mechanisms. In particular, given a description of the domain of the available courses, the reasoning capabilities provided by DyLOG have been exploited both to guide a student in building a study plan to acquire some desired competence, and to verify whether a study plan proposed by a student is correct, e.g., course preconditions are respected and the plan allows the student to acquire the desired competence. Indeed, verifying the correctness of a user-given study plan can be naturally interpreted in the DyLOG framework as an instance of the temporal projection problem, where, given a sequence of courses (actions) a_1, \dots, a_n composed by the user and a set of competences represented by the fluent conjunction Fs , we want to verify if the sequence is a correct study plan for achieving the competence Fs (the learning goal). Moreover, the problem of constructing a study plan that achieves a learning goal and user’s conditions, and fits in a predefined curriculum schemas can be naturally interpreted as a planning problem à la DyLOG, when we represent curriculum schemas as procedures and we look for a possible execution that lead to satisfy the goal.

Example 5.1. To give a more concrete flavor, let us see the top level procedure that specify the behavior of a reasoner when it is requested to perform a study plan construction task. It is called *advice* and mainly extract a study plan that after will be executed (proposed to the student via HTML pages).

$$\begin{aligned} & \text{advice}(\text{Plan}) \text{ is} \\ & \text{ask_user_preferences} \wedge ?\text{requested}(\text{Curriculum}) \wedge \\ & \text{plan}(\text{credits}(C) \wedge \text{max_credits}(B) \wedge (C \leq B) \text{ after} \\ & \text{achieve_goal}(\text{has_competence}(\text{Curriculum}), \text{Plan}) \wedge \text{Plan}. \end{aligned}$$

Intuitively, the reasoner asks the student what kind of final expertise he wants to achieve and his background knowledge (e.g., if he already attended some of the possible courses). Afterwards, it adopts the user’s goals and tries to build a plan for achieving

the goal of having the desired final expertise. The *meta-predicate plan* returns the plan, in this case by extracting those executions of the procedure *achieve_goal* that satisfy the user's goals as well as the further conditions that are possibly specified (e.g., that the number of credits gained by following the study plan is not bigger than a predefined maximum). The extracted plan can be *conditional*, predicting also the future interactions with the user. In fact, if the agent finds different courses that supply a same competence, whose prerequisites are satisfied, it is programmed in such a way that it asks the user to make a choice. The conditional plan returned by the reasoning process can be executed in the on-line modality. Every *action* has some *code* associated to it, that is to be performed when the action is actually executed; such a code produces the real effects of the action in the world, that in our case consists in sending to the executor the message of showing a given web page.

In the procedure *advice* above the agent behavior starts by interacting with the student, that is mainly asked to specify which kind of support he needs and his learning goals (*ask_user_preferences*). In DyLOG the *interaction with the user* is specified by means of sensing actions. In fact, generally sensing actions allow an agent to gather inputs from the *external world*, thus, in the web application context, it seemed to be a natural choice to use sensing for requesting *the user* to enter a value for a fluent, true or false in case of ordinary fluents, a value from the domain in case of fluents with a finite domain.

In the simplest case the user is explicitly requested to enter a truth value for a fluent. This kind of interaction, however is not sufficient, because rather than asking for a fluent truth value, it can be useful to offer a set of alternatives to the user, among which he will make a choice. To deal with this matter, in [4] DyLOG has been augmented by introducing a special subset of sensing actions, called *suggesting actions* which can be used when the agent has to find out the value of fluents representing the *user's preferences* among a finite subset of alternatives. The difference w.r.t. standard sensing actions is that while those consider as alternative values for a given fluent its whole domain, suggesting actions offer only a subset of it. The agent has an active role in selecting the possible values among which the user chooses: only those that lead to fulfill the goal will be selected. Such values are identified by means of a reasoning process. Such difference arises when we cope with conditional plan extraction. In the normal sensing case the agent *must* consider all the possible outcomes of sensing, then it looks for a conditional plan that succeeds for all the possible input values. Instead suggesting actions allow the agent to reason about the options it could offer to the user and to *select* only the ones that lead to fulfill certain goals (the idea is that only the selected options will be offered at execution time). Formally, for modeling this kind of reasoning, it is sufficient to slightly modify the rule 4-bis of proof procedure for generating conditional plans presented in section 4.2, by defining \mathcal{F} in the following way:

$$\mathcal{F} = \max(\mathcal{F}') \quad \text{s.t.} \quad \mathcal{F}' \subseteq \{l_1, \dots, l_t\} = \text{dom}(s).$$

Basically, when a suggesting action s is executed, the proof procedure can consider, instead of the whole associated domain $dom(s)$ of the possible outcomes, the biggest subset of it for which the computation (split in branches) succeeds.

6. Related work

The problem of reasoning about actions in presence of sensing and of incomplete states has been tackled by many authors in the literature. In the Scherl and Levesque' work [40] a framework has been proposed to formalize knowledge-producing actions in classical logic, adapting the possible world model of knowledge to the situation calculus. As a difference, we describe an epistemic state by a set of epistemic literals, a simplification similar to the one considered in [13], which leads to a loss of expressivity, but to a gain in tractability.

In [33] Levesque formulates the planning task in domains including sensing. Starting from the theory of sensing actions in situation calculus presented in [40], he defines complex plans as robot programs, that may contain sensing actions, conditionals and loops, and specifies the planning task as the problem to find a robot program achieving a desired goal from a certain initial state. However the paper does not suggest how to generate automatically such robot plans, while we presented a proof procedure to deal with it (section 4.2). On the same line there is a recent paper by De Giacomo et al. [28] where some desired properties of a program returned by a planning process and ready to be executed have been investigated. The framework of reference is IndiGolog, a variant of GOLOG intended to be executed on-line in an incremental way. In particular the authors propose a formal characterization of plans as *epistemically feasible programs*, i.e. programs for which an executing agent, at every stage of execution, by virtue of what it knew initially and the subsequent reading of sensors, always can decide what step to take next toward the goal. Based on this notion, they considered two kind of programs restricted w.r.t. the syntactic form, namely linear programs (programs that not performs sensing) and *tree programs* (roughly, conditional plans where one can only test a condition that has just been sensed). It can be proved that when a program belonging to one of these classes is executable, it is also epistemically feasible, i.e. the agent always knows what to do next. Notice that the conditional plans that are extracted by our proof procedure have the form of tree programs.

The works in [13,14,35] have tackled the problem of extending the Gelfond and Lifschitz' language \mathcal{A} for reasoning about complex plans in presence of sensing and incomplete information. In [35] Lobo et al. introduce the language \mathcal{A}_K , which provides both actions to increase agent knowledge and actions to lose agent knowledge. It has a general semantics in which epistemic states are represented by sets of worlds. The epistemic state of an agent is represented in \mathcal{A}_K as a set of worlds (states), rather than by a set of epistemic literals as in our proposal. This makes the semantics of \mathcal{A}_K more general than ours. In particular, \mathcal{A}_K disjunctive knowledge can be represented, while it cannot in our approach. However, since in the language of \mathcal{A}_K there is not an explicit representation of epistemic fluents, it is not possible for the agent to query itself

about its knowledge (the agent has no introspection). Precondition laws to rule executability of actions are not provided. In particular, knowledge laws, which describe the effects of sensing actions, have preconditions on the effects of actions rather than on their executability. In \mathcal{A}_K complex plans are defined as Algol-like programs, containing sequence, conditional statements and iteration. Given a domain description in \mathcal{A}_K , a query of the form ϕ **after** $[\alpha]$ is true if ϕ holds in every model of D after the execution of the plan α in the initial state, where α is a complex plan, possibly including conditionals and iterations. As a difference with [35], rather than verifying the correctness of a plan, in this paper we have addressed the problem of finding a finite conditional plan (a possible execution of a procedure) which is provably correct with respect to a given condition.

In [14] Baral and Son define an action description language, also called \mathcal{A}_K , which deals with sensing actions and distinguishes between the state of the world and the state of knowledge of an agent about the world. The semantics of the language is proved to be equivalent to the one in [35] when rational models are considered. Baral and Son [13,14] define several sound approximation of the language \mathcal{A}_K with a smaller state space with respect to \mathcal{A}_K , based on three-valued interpretations. Our approach has strong similarities with the 0-approximation. Indeed, our epistemic states are, essentially, three-valued models and, as for the 0-approximation, our language does not provide reasoning about cases. The meaning of queries in [14] is substantially similar to the one in [35] and, therefore, it is different from ours.

Following [17,18], in [22] De Giacomo and Rosati propose a minimal knowledge approach for reasoning about actions and sensing, in presence of incomplete information. They use a formalism which combines the modal μ -calculus and autoepistemic logic. In their language, they have epistemic formulas $\mathbf{k}p$ (where p is a literal conjunction) and they allow precondition laws of the form $\mathbf{k}p \supset \langle a \rangle true$ and action laws of the form $\mathbf{k}p \supset [a]\mathbf{k}q$. On the other hand, their domain description does not contain formulas of the form $\mathcal{M}p \supset [a]\mathcal{M}q$, which in our case are needed for describing the possible effects of an action when there is uncertainty about its preconditions. The last kind of action laws in our approach are also used for modelling actions with non-deterministic effects, which make the agent to loose information. Such actions are not provided in [22]. An algorithm is introduced to compute a transition graph from an action specification. This graph can be used for verifying properties of the possible executions through model checking and to prove rather sophisticated temporal properties like liveness and maintenance goals. Though sensing actions are specified as nondeterministic actions, their treatment in the construction of the transition graph is similar to ours, in that, a sensing action is regarded as the nondeterministic choice of two atomic actions, the one which makes the fluent known, and the other one which make its negation known. Frame axioms are only provided for sensing actions, and in a way that a sensing action does not have any effect on fluents whose value is known before their execution.

In [42] Thielscher faces the problem of representing a robot's knowledge about its environment in the context of the Fluent Calculus, a formalism for reasoning about ac-

tions based on predicate logic. In order to account for knowledge, basic fluent calculus is extended by introducing the concept of possible world state and defining knowledge of a robot in terms of possible states. The formalism deals with sensing actions and it allows to distinguish between state of the world and state of knowledge of an agent about the world. A monotonic solution to the frame problem for knowledge is provided, by means of suitable knowledge update axioms but, as a difference with [14], independent specifications of state and knowledge update can be given. A concept of conditional action, denoted by $If(f, a)$, is introduced in order to deal with planning in presence of sensing. Such If -constructs allow the robot to condition its course of actions on the result of sensing actions included in its plan. However If -constructs uses only atomic conditions, while our formalism allow to express as complex actions conditional constructs with arbitrary complex conditions.

In [37] Petrick and Bacchus present an approach to planning with incomplete information and sensing, where planners states are represented as sets of formulas from a modal logic of knowledge. As in our approach, actions are modeled in terms of how they modify the knowledge state of the planner rather than in terms of how they modify the physical world. Formulas in the knowledge state can be first-order modal formulas, and, in order to retain tractability on the kind of reasoning which can be performed, the knowledge state is structured as a collection of four databases, and queries are suitable restricted.

As concerns the problem of defining complex actions, there is a close relation between our language and GOLOG [34], though, from the technical point of view, it is based on a different approach. While our language makes use of modal logic, GOLOG is based on classical logic and, more precisely, on the situation calculus. We make use of abduction to deal with persistency, while in GOLOG is given a monotonic solution of the frame problem by introducing successor state axioms. In our case, procedures are defined as axioms of our modal logic, while in GOLOG they are defined by macro expansion into formulae of the situation calculus. GOLOG definition is very general, but it makes use of second order logic to define iteration and procedure definition. Hence there is a certain gap between the general theory on which GOLOG is based and its implementation in Prolog. In contrast, we have tried to keep the definition of the semantics of the language and of its proof procedure as close as possible.

We would like to mention the work in [20,21] where GOLOG is extended in order to deal with concurrency (ConGolog) and with *execution* of high-level programs. In particular, in [21] the problem is tackled of executing programs including sensing. The notions of *off-line* and *on-line* execution are introduced and a way of combining them is considered.

Between the other approaches addressing the problem to reason about complex actions, let us mention the work of Hölldobler et al. [31], where the focus is on the definition of a planning language for specifying complex plans. This language, based on first-order logic, allows for procedure definitions, conditional and recursive plans, and some form of non-deterministic choice. The authors give also a formal definition of the notions of executability, termination and correctness of complex plans, from the point of

view of a skeptical agent. Nevertheless the problem of treating sensing is not addressed and it is possible to deal with uncertain knowledge only by using non-determinism.

7. Conclusions

In this paper we have presented a logic programming language for modelling and programming rational agents. The language is based on a modal theory of actions and mental attitudes where modalities are used for representing actions and beliefs modelling the agent's mental state. Our action theory allows to deal with sensing actions as well as with complex actions. The problem of reasoning about complex actions with incomplete knowledge has been tackled and in particular the temporal projection and planning problem have been addressed. We adopted a non-monotonic approach to deal with the frame problem, by making use of abduction: persistency is achieved by maximizing persistency assumptions. The integration of non-monotonic techniques and modal logics leads to a very expressive and powerful language that allows to reason about action and change, as well as to model the agent's mental attitude dynamics. Moreover, the adoption of the *logic programming paradigm* is crucial in order to define a language which can be used both for specifying and for programming agents, and then to bridge the gap between logical model and practical implementation of agent systems.

In this paper we also briefly motivated and describe our experience in using DyLOG for implementing web applications, where reasoning and planning techniques are used for supplying adaptive services to users and we refer to [4,9] for more details.

In the last few years, the AI community devoted a great deal of attention to the issue of communication and dialogue among agents in the context of a formal approach to the theory of agency [24,26,39]. On this line in a set of works [6,7,36] the problem of providing a DyLOG agent with a *communication kit* has been tackled. The logical action framework has been extended to integrate a communication theory and the problem of specifying and reasoning about communications and conversation protocols has been faced.

The formal account of communication we proposed aims at coping with two main aspects: the state change caused by a communicative act on the local agent's mental state, and the decision strategy used by an agent for sending suitable answers to a received communication. Regarding the first aspect, the DyLOG action theory has been extended to represent primitive communicative actions in terms of preconditions and effects on mental states, including (possibly nested) belief and goal fluents. Regarding the second aspect, agents have been equipped with a set of FIPA-like conversation protocols, modelled by taking the agent's point of view and by building on primitive speech acts. Such protocols specify possible communication patterns for agent's conversations and then guide the selection process of the proper answer, constraining the search space of possible agent responses. The communication theory is viewed as a homogeneous component of the general agent theory, as both conversational policies, that guide the agent's communicative behavior, and other policies defining the agent's complex behavior are represented by non-deterministic procedures definitions (procedure axioms). The proof

procedure presented in section 4 has been adapted to the extended framework, in order to support agent's reasoning and planning in presence of communications.

In [5] we have shown how to interpret the semantic web, and in particular web services, in our framework for multiagent communication. A web service can be seen as an agent which communicates with other agents, and the behavior of the service can be expressed as a conversation protocol in a logic language. Having a logic specification of the protocol, it is possible to reason about the effects of engaging specific conversations, and to verify properties of the protocol.

Acknowledgements

We are grateful to the anonymous referees for their careful reading of the paper, their useful suggestions and constructive criticisms.

Appendix A. Soundness results

The proof of soundness of the proof procedure in section 4.1 with respect an acceptable solution makes use of a soundness and completeness result for the monotonic part of the proof procedure w.r.t. the monotonic part of the semantics. Indeed, if the assumptions $\mathbf{M}[a_1; \dots; a_m]F$ are regarded as facts rather than abducibles and they are added to the program, the non-monotonic rule 7c in the proof procedure can be replaced by a monotonic one. The resulting monotonic proof procedure can be shown to be sound and complete with respect to the Kripke semantics of the modal logics $\mathcal{L}_{(\Pi, S_0)}$. Formally, the monotonic proof procedure is defined by the relation \vdash_{Δ} .

Definition A.1. Let Δ be a consistent set of abductive assumptions. The relation \vdash_{Δ} is defined by the rules 1–7b, 7d–9 in figures 4 and 5 in section 4.1, where \vdash is replaced with \vdash_{Δ} , and by the following rule:

$$7c') \quad \overline{a_1; \dots; a_m \vdash_{\Delta} F} \quad \text{where } \mathbf{M}[a_1, \dots, a_m]F \in \Delta.$$

We will write $a_1, \dots, a_m \vdash_{\Delta} \langle p_1; p_2; \dots; p_n \rangle Fs$ with answer σ to mean that the query $\langle p_1; p_2; \dots; p_n \rangle Fs$ can be proved from the domain description (Π, S_0) at the state a_1, \dots, a_m with answer σ by means of rules for the monotonic proof procedure defined above.

Theorem A.1 (Soundness of \vdash_{Δ}). Let (Π, S_0) be an e-consistent dynamic domain description and let G be a query of form $\langle p_1; p_2; \dots; p_n \rangle Fs$. Let Δ be a consistent set of abductive assumptions. If there is a \vdash_{Δ} -proof Υ for a query G from a dynamic domain description (Π, S_0) at state a_1, \dots, a_m , then $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m]G$.

Proof. We prove the soundness of \vdash_{Δ} by induction on the height of the proof Υ . If the height h of Υ is 1, then Υ is an axiom. There are four cases, rules 6, 7b, 7c', 7d, and for

all of them the theorem holds trivially. By inductive hypothesis the theorem holds for queries whose proof Υ has height less than or equal to h . Let us prove it for $h + 1$. We consider the following cases, one for each inference rule in which Υ can terminate.

Case rule 1. Assume that the root inference figure in Υ is rule 1. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1 \quad a_1, \dots, a_m \vdash_{\Delta} \langle p'_1; \dots; p'_{n'}; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{\Delta} \langle p; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $p \in \mathcal{P}$ and $\langle p \rangle \varphi \subset \langle p'_1; \dots; p'_{n'} \rangle \varphi \in \Pi_{\mathcal{P}}$. Let $G' = \langle p_2; \dots; p_n \rangle Fs$. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m] \langle p \rangle G'$. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m] \langle p \rangle G'$ holds.

Since Υ_1 is shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m] \langle p'_1; \dots; p'_{n'} \rangle G'$. Now, since we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we have that for each interpretation M and for each world w , $M, w \models [a_1; \dots; a_m] \langle p'_1; \dots; p'_{n'} \rangle G'$. Then, by definition of satisfiability, we have $M, w' \models \langle p'_1; \dots; p'_{n'} \rangle G'$, for any world w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$. Since $\langle p'_1; \dots; p'_{n'} \rangle \varphi \supset \langle p \rangle \varphi$ is an axiom in $\Pi_{\mathcal{P}}$, we have also $M, w' \models \langle p'_1; \dots; p'_{n'} \rangle G' \supset \langle p \rangle G'$ and hence $M, w' \models \langle p \rangle G'$. Since it holds for any w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$, it follows that $M, w \models [a_1; \dots; a_m] \langle p \rangle G'$, for any interpretation M and world w , that is $\models [a_1; \dots; a_m] \langle p \rangle G'$.

Case rule 2. Assume that the root inference figure in Υ is rule 2. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1 \quad a_1, \dots, a_m \vdash_{\Delta} Fs' \quad \Upsilon_2 \quad a_1, \dots, a_m \vdash_{\Delta} \langle p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{\Delta} \langle (Fs')?; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

Let $G' = \langle p_2; \dots; p_n \rangle Fs$. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m] \langle (Fs')? \rangle G'$. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m] \langle (Fs')? \rangle G'$ holds.

Since Υ_1 and Υ_2 are shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m] Fs'$ and $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m] G'$ hold. Now, since we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we have that for each interpretation M and for each world w , $M, w \models [a_1; \dots; a_m] Fs'$ and $M, w \models [a_1; \dots; a_m] G'$. Then, by definition of satisfiability, we have (1) $M, w' \models Fs'$ and (2) $M, w' \models G'$, for any w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$. From (1) and (2), by definition of satisfiability, we have also $M, w' \models \langle (Fs')? \rangle G'$ and, since it holds for any w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$, it follows that $M, w \models [a_1; \dots; a_m] \langle (Fs')? \rangle G'$, for any interpretation M and world w , that is $\models [a_1; \dots; a_m] \langle (Fs')? \rangle G'$.

Case rule 3. Assume that the root inference figure in Υ is rule 3. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1 \quad \Upsilon_2}{a_1, \dots, a_m \vdash_{\Delta} \langle a; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma} \quad \frac{a_1, \dots, a_m \vdash_{\Delta} Fs' \quad a_1, \dots, a_m, a \vdash_{\Delta} \langle p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{\Delta} \langle a; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $a \in \mathcal{A}$ and $\Box(Fs' \supset \langle a \rangle \top) \in \Pi_{\mathcal{A}}$. Let $G' = \langle p_2; \dots; p_n \rangle Fs$. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m] \langle a \rangle G'$. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m] \langle a \rangle G'$ holds.

Since Υ_1 and Υ_2 are shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m] Fs'$ and $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m, a] G'$ hold, with $\Box(Fs' \supset \langle a \rangle \top) \in \Pi_{\mathcal{A}}$ in Π . Thus, since we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then $\models [a_1; \dots; a_m] Fs'$ and $\models [a_1; \dots; a_m; a] G'$ and also $\models \Box(Fs' \supset \langle a \rangle \top)$, since the law belong to $\Pi_{\mathcal{A}}$ in Π and we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$. By definition of validity respect to the class $\mathcal{M}_{(\Pi, Obs)}$, it means that for each Kripke interpretation M and for each world $w \in W$, $M, w \models [a_1; \dots; a_m] Fs'$, $M, w \models [a_1; \dots; a_m; a] G'$ and $M, w \models \Box(Fs' \supset \langle a \rangle \top)$. In particular, for definition of satisfiability, $M, w \models [a_1; \dots; a_m] Fs'$ iff $M, w' \models Fs'$, for each world w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$. Since $\mathcal{R}_{a_1; \dots; a_m} \subseteq \mathcal{R}_{\Box}$, then $M, w' \models Fs' \supset \langle a \rangle \top$ that together with $M, w' \models Fs'$ implies $M, w' \models \langle a \rangle \top$. By satisfiability definition, it means that it exists a world w'' s.t. $(w', w'') \in \mathcal{R}_a$. Since $M, w \models [a_1, \dots, a_m, a] G'$ and $(w, w'') \in \mathcal{R}_{a_1; \dots; a_m; a}$, then $M, w'' \models G'$. From $M, w' \models \langle a \rangle \top$ and $M, w'' \models G'$ it follows that $M, w' \models \langle a \rangle G'$. Since it holds for each w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$, we have that, for each interpretation M , for each w , $M, w \models [a_1; \dots; a_m] \langle a \rangle G'$, that is $\models [a_1; \dots; a_m] \langle a \rangle G'$.

Case rule 4. Assume that the root inference figure in Υ is rule 4. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1}{a_1, \dots, a_m \vdash_{\Delta} \langle s^{Bl}; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma} \quad \frac{a_1, \dots, a_m \vdash_{\Delta} \langle s^{Bl}; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{\Delta} \langle s; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $s \in \mathcal{S}$ and $l \in \text{dom}(s)$. Let $G' = \langle p_2; \dots; p_n \rangle Fs$. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m] \langle s \rangle G'$, where s is a sensing action in \mathcal{S} . Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m] \langle s \rangle G'$ holds.

Since Υ_1 is shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m] \langle s^{Bl} \rangle G'$ holds, with $l \in \text{dom}(s)$. Since we assumed $\Sigma_{(\Pi, S_0)} \cup \Delta$ we have that for each interpretation M and for each world w , $M, w \models [a_1; \dots; a_m] \langle s^{Bl} \rangle G'$, that, by satisfiability definition, implies $M, w' \models \langle s^{Bl} \rangle G'$, for each w' such that $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$. From $M, w' \models \langle s^{Bl} \rangle G'$, by definition of satisfiability, we have that it exists a world w'' s.t. $(w', w'') \in \mathcal{R}_{s^{Bl}}$ and $M, w'' \models G'$. Moreover, since $s \in \mathcal{S}$, $[s]\varphi \equiv [\bigcup_{l \in \text{dom}(s)} s^{Bl}]\varphi$ is an axiom in $\Pi_{\mathcal{S}}$. Therefore the inclusion property between accessibility relations $\mathcal{R}_s \supseteq \bigcup_{l \in \text{dom}(s)} \mathcal{R}_{s^{Bl}}$ holds. Hence, from the fact that it exists w'' s.t. $(w', w'') \in \mathcal{R}_{s^{Bl}}$ and $M, w'' \models G'$, we can conclude that (w', w'') belongs to the relation \mathcal{R}_s too and, by definition of satisfiability, that $M, w' \models \langle s \rangle G'$ holds. Since it

holds for each w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$, we have that, for each interpretation M , for each w , $M, w \models [a_1; \dots; a_m]\langle s \rangle G'$, that is $\models [a_1; \dots; a_m]\langle s \rangle G'$.

Case rule 5. Assume that the root inference figure in Υ is rule 5. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1}{a_1, \dots, a_m \vdash_{\Delta} Fs} \frac{}{a_1, \dots, a_m \vdash_{\Delta} \langle \varepsilon \rangle Fs \text{ w.a. } \sigma}$$

where $\sigma = a_1; \dots; a_m$. Obvious, by inductive hypothesis.

Case rule 7a. Assume that the root inference figure in Υ is rule 7a. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1}{a_1, \dots, a_{m-1} \vdash_{\Delta} Fs'} \frac{}{a_1, \dots, a_m \vdash_{\Delta} F}$$

where $m > 0$ and $\Box(Fs' \supset [a_m]F) \in \Pi_{\mathcal{A}}$. To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m]F$, where F is an epistemic fluent. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m]F$ holds. Since Υ_1 is shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_{m-1}]Fs'$ holds, with $\Box(Fs' \supset [a_m]F) \in \Pi_{\mathcal{A}}$ in Π . Thus, since we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we have that for each interpretation M and world w , $M, w \models [a_1; \dots; a_{m-1}]Fs'$ and also $M, w \models \Box(Fs' \supset [a_m]F)$. In particular, for definition of satisfiability, $M, w \models [a_1; \dots; a_{m-1}]Fs'$ iff $M, w' \models Fs'$, for each world w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_{m-1}}$. Since $\mathcal{R}_{a_1; \dots; a_{m-1}} \subseteq \mathcal{R}_{\Box}$, then $M, w' \models Fs' \supset [a_m]F$ that together with $M, w' \models Fs'$ implies $M, w' \models [a_m]F$. Since it holds for any w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_{m-1}}$, it follows that $M, w \models [a_1; \dots; a_{m-1}][a_m]F$, for any interpretation M and world w , i.e. $\models [a_1; \dots; a_m]F$.

Case rule 8. Assume that the root inference figure in Υ is rule 8. Hence Υ has form:

$$\frac{\Upsilon_1 \quad \Upsilon_2}{a_1, \dots, a_m \vdash_{\Delta} Fs_1 \quad a_1, \dots, a_m \vdash_{\Delta} Fs_2} \frac{}{a_1, \dots, a_m \vdash_{\Delta} Fs_1 \wedge Fs_2}$$

Since Υ_1 and Υ_2 are shorter than Υ , by inductive hypothesis we have $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m]Fs_1$ and $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m]Fs_2$ and hence, for definition of satisfiability relation, $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m]Fs_1 \wedge Fs_2$.

Case rule 9. Assume that the root inference figure in Υ is rule 9. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon_1}{a_1, \dots, a_m \vdash_{\Delta} \mathcal{B}l} \frac{}{a_1, \dots, a_m \vdash_{\Delta} \neg \mathcal{B} \neg l}$$

To prove the thesis, we prove $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m] \neg \mathcal{B} \neg l$. Assume $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we show that $\models [a_1; \dots; a_m] \neg \mathcal{B} \neg l$ holds.

Since Υ_1 is shorter than Υ , by inductive hypothesis, we get that $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m]\mathcal{B}l$. Now, since we assumed $\models \Sigma_{(\Pi, S_0)} \cup \Delta$, then we have that for each interpretation M and for each world w , $M, w \models [a_1; \dots; a_m]\mathcal{B}l$. Then, by definition of satisfiability, we have $M, w' \models \mathcal{B}l$, for any world w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$. Moreover, since modality \mathcal{B} is serial and $\mathcal{B}\varphi \supset \neg\mathcal{B}\neg\varphi$ is an axiom of our modal logic, we have $M, w' \models \mathcal{B}l \supset \neg\mathcal{B}\neg l$. By modus ponens it follows that $M, w' \models \neg\mathcal{B}\neg l$. Since it holds for any w' s.t. $(w, w') \in \mathcal{R}_{a_1; \dots; a_m}$, it follows that $M, w \models [a_1; \dots; a_m]\neg\mathcal{B}\neg l$, for any interpretation M and world w , i.e. $\models [a_1; \dots; a_m]\neg\mathcal{B}\neg l$. \square

Let us consider now the *completeness* of the monotonic proof procedure \vdash_{Δ} respect to the Kripke semantics. The completeness proof is given by constructing a canonical model for a given theory $\Sigma_{(\Pi, S_0)} \cup \Delta$, where $\Sigma_{(\Pi, S_0)}$ is a theory in $\mathcal{L}_{(\Pi, S_0)}$ and Δ is a given consistent set of abductive assumptions. We introduced the notion of operational derivability of a query G from a domain description (Π, S_0) in a certain state represented by the sequence a_1, \dots, a_m . A sequence a_1, \dots, a_m , with $m \geq 0$, is a shorthand for the sequence of modalities $[a_1] \dots [a_m]$, where the a_i 's are world actions, and it keeps track of the sequence of actions performed during the computation. In the following, we will refer to such sequences as *modal contexts*. Intuitively, a modal context is a name for a possible world. During the model's construction, we will use modal contexts, possibly extended with epistemic modal operators, in order to specify the set W of possible words of the model. We will denote by ε the empty sequence of modalities.

Definition A.2 (Derivation relation $\overset{*}{\Rightarrow}_{\Pi_{\mathcal{P}}}$). Given a set $\Pi_{\mathcal{P}}$ of procedure axioms in a domain description (Π, S_0) , the derivation relation $\overset{*}{\Rightarrow}_{\Pi_{\mathcal{P}}}$ is the transitive and reflexive closure of the relation $\Rightarrow_{\Pi_{\mathcal{P}}}$ defined as follow: for each $[p_0]\varphi \supset [p_1][p_2] \dots [p_n]\varphi \in \Pi_{\mathcal{P}}$ and for each modal context $\Gamma, \Gamma', \Gamma p_0 \Gamma' \Rightarrow_{\Pi_{\mathcal{P}}} \Gamma p_1 \dots p_n \Gamma'$.

Definition A.3 (Canonical model). The canonical model \mathcal{M}_c for a theory $T = \Sigma_{(\Pi, S_0)} \cup \Delta$, where (Π, S_0) is an e-consistent domain description, is a tuple

$$\langle W, \mathcal{R}_{\mathcal{B}}, \{\mathcal{R}_{a_i}: a_i \in \mathcal{A}\}, \{\mathcal{R}_s: s \in \mathcal{S}\}, \{\mathcal{R}_p: p \in \mathcal{P}\}, \mathcal{R}_{\square}, V \rangle$$

where:

- $W = \{a_1 \dots a_n \mathcal{B}, a_1 \dots a_n \mathcal{M}, a_1 \dots a_n: n \geq 0, a_i \in \mathcal{A}\}$ where $a_1 \dots a_n \mathcal{B} (a_1 \dots a_n \mathcal{M})$ denote the concatenation between the modal context $a_1 \dots a_n$ and the operator $\mathcal{B} (\mathcal{M})$;
- $\mathcal{R}_{a_i} = \{(a_1 \dots a_n, a_1 \dots a_n a_i) \in W \times W: a_1, \dots, a_n \vdash_{\Delta} F s', \text{ with } \square(F s' \supset \langle a_i \rangle \top) \in \Pi\}$;
- \mathcal{R}_{\square} is a binary relation on $W \times W$. It is reflexive, transitive, and satisfies the condition $\mathcal{R}_{\square} \supseteq (\bigcup_{a_i} \mathcal{R}_{a_i})^*$, $a_i \in \mathcal{A}$, i.e. \mathcal{R}_{\square} contains the reflexive and transitive closure of the union of the \mathcal{R}_{a_i} ;
- $\mathcal{R}_p = \{(a_1 \dots a_n, a_1 \dots a_n a_{n+1} \dots a_{n+m}) \in W \times W: p \overset{*}{\Rightarrow}_{\Pi_{\mathcal{P}}} a_{n+1} \dots a_{n+m}\}$;

- $\mathcal{R}_s = \{(a_1 \dots a_n, a_1 \dots a_n s^F) \in W \times W : s \in \mathcal{S}, s^F \in \mathcal{A}, \text{ where } F \text{ is an epistemic fluent};$
- $\mathcal{R}_B = \{(a_1 \dots a_n, a_1 \dots a_n \mathcal{B}) \in W \times W\} \cup \{(a_1 \dots a_n, a_1 \dots a_n \mathcal{M}) \in W \times W\} \cup \{(a_1 \dots a_n \mathcal{B}, a_1 \dots a_n \mathcal{B}) \in W \times W\} \cup \{(a_1 \dots a_n \mathcal{M}, a_1 \dots a_n \mathcal{M}) \in W \times W\};$
- for each fluent name f , for each modal context $a_1 \dots a_n, n \geq 0$ we set:
 - (a) $V(a_1 \dots a_n \mathcal{B}, f) = \mathbf{T}$ iff $a_1, \dots, a_n \vdash_{\Delta} \mathcal{B}f$;
 - (b) $V(a_1 \dots a_n \mathcal{M}, l) = \mathbf{T}$ iff $a_1, \dots, a_n \vdash_{\Delta} \neg \mathcal{B} \neg f$ and $a_1, \dots, a_n \vdash_{\Delta} \neg \mathcal{B}f$, or $a_1, \dots, a_n \vdash_{\Delta} \mathcal{B}f$.

We recall back that in the monotonic formulation abducibles are considered to be new atomic propositions. Then, we set:

- (c) $V(\varepsilon, \mathbf{M}[a_1; \dots; a_m]F) = \mathbf{T}$ iff $\mathbf{M}[a_1; \dots; a_m]F \in \Delta$.

In all other cases the function V is defined to be false.

Moreover, we have:

- $\mathcal{R}_{\psi?} = \{(a_1 \dots a_n, a_1 \dots a_n) \in W \times W : a_1, \dots, a_n \vdash_{\Delta} \psi\}$.

The canonical model \mathcal{M}_c for a theory $\Sigma_{(\Pi, S_0)} \cup \Delta$ given by definition A.3 is a Kripke (Π, S_0) -interpretation. In fact, it is easy to see that each property on accessibility relations stated in definition 3.1 is satisfied by the canonical model \mathcal{M}_c .

The following proposition states that the canonical model \mathcal{M}_c for a theory $\Sigma_{(\Pi, S_0)} \cup \Delta$ is a Kripke (Π, S_0) -interpretation.

Proposition A.1. The canonical model \mathcal{M}_c given by definition A.3 is a Kripke (Π, S_0) -interpretation, that is:

- (1) each property on accessibility relations stated in definition 3.1 is satisfied by the canonical model \mathcal{M}_c , and
- (2) V is a valuation function.

Proof. (1) The proof is omitted. It is based on quite standard techniques and it similar to the ones presented in [2].

(2) The proof is based on standard techniques [32] and it is omitted. Let us stress that, in order to prove that the value-assignment V of \mathcal{M}_c in definition A.3 is a valuation function, it is essential the e-consistency requirement on the domain description (Π, S_0) . \square

Completeness proof is based on the following two properties of \mathcal{M}_c .

Theorem A.2. Let $\Sigma_{(\Pi, S_0)}$ be a theory in $\mathcal{L}_{(\Pi, S_0)}$, where (Π, S_0) is an e-consistent domain description, let be Δ a consistent set of abductive assumptions, and \mathcal{M}_c the canonical model of $\Sigma_{(\Pi, S_0)} \cup \Delta$. Let G be a query of form $\langle p_1; p_2; \dots; p_n \rangle Fs$, then the following properties hold:

- (1) for each modal context $a_1 \dots a_n : a_{i=1, \dots, n} \in \mathcal{A}$,
 $\mathcal{M}_c, a_1 \dots a_n \models G$ iff $a_1, \dots, a_n \vdash_\Delta G$;
- (2) \mathcal{M}_c satisfies $\Sigma_{(\Pi, S_0)} \cup \Delta$; i.e., since $\Sigma_{(\Pi, S_0)} \cup \Delta = \{\Pi_{\mathcal{A}} \cup \Delta \cup S_0\}$, for all formulae
 $D \in \{\Pi_{\mathcal{A}} \cup \Delta \cup S_0\}$, $\mathcal{M}_c, \varepsilon \models D$.

Proof. We prove property (1) by induction on the structure of G .

- $G = T$: trivial.
- $G = F$: there are four cases. F have form $\mathcal{B}f$, $\mathcal{B}\neg f$, $\neg\mathcal{B}f$, or $\neg\mathcal{B}\neg f$.

$G = \mathcal{B}f$: $\mathcal{M}_c, a_1 \dots a_n \models \mathcal{B}f$ iff for each world w' s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{\mathcal{B}}$, $\mathcal{M}_c, w' \models f$. Now, by definition of $\mathcal{R}_{\mathcal{B}}$, iff $\mathcal{M}_c, a_1 \dots a_n \mathcal{B} \models f$ and $\mathcal{M}_c, a_1 \dots a_n \mathcal{M} \models f$, that is, iff (1) $V(a_1 \dots a_n \mathcal{B}, f) = \mathbf{T}$ and (2) $V(a_1 \dots a_n \mathcal{M}, f) = \mathbf{T}$. By definition of V in \mathcal{M}_c , iff (1) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$ and (2) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$, or $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$; that is, iff $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$

$G = \mathcal{B}\neg f$: $\mathcal{M}_c, a_1 \dots a_n \models \mathcal{B}\neg f$ iff for each world w' s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{\mathcal{B}}$, $\mathcal{M}_c, w' \models \neg f$. Now, by definition of $\mathcal{R}_{\mathcal{B}}$, iff $\mathcal{M}_c, a_1 \dots a_n \mathcal{B} \models \neg f$ and $\mathcal{M}_c, a_1 \dots a_n \mathcal{M} \models \neg f$, that is, iff (1) $V(a_1 \dots a_n \mathcal{B}, \neg f) = \mathbf{F}$ and (2) $V(a_1 \dots a_n \mathcal{M}, \neg f) = \mathbf{F}$. By definition of V in \mathcal{M}_c , iff (1) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$, or $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$, and (2) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$; that is, iff $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$.

$G = \neg\mathcal{B}f$: $\mathcal{M}_c, a_1 \dots a_n \models \neg\mathcal{B}f$ iff it exists a world w' s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{\mathcal{B}}$ and $\mathcal{M}_c, w' \models \neg f$. Now, by definition of $\mathcal{R}_{\mathcal{B}}$, iff $\mathcal{M}_c, a_1 \dots a_n \mathcal{B} \models \neg f$ or $\mathcal{M}_c, a_1 \dots a_n \mathcal{M} \models \neg f$, that is, iff (1) $V(a_1 \dots a_n \mathcal{B}, \neg f) = \mathbf{F}$ or (2) $V(a_1 \dots a_n \mathcal{M}, \neg f) = \mathbf{F}$. By definition of V in \mathcal{M}_c , iff (1) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$, or $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$, or (2) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$. If $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$ is the case, it is obvious, otherwise if $a_1, \dots, a_n \vdash_\Delta \mathcal{B}\neg f$ is the case, we obtain the thesis by applying the rule 9.

$G = \neg\mathcal{B}\neg f$: $\mathcal{M}_c, a_1 \dots a_n \models \neg\mathcal{B}\neg f$ iff it exists a world w' s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{\mathcal{B}}$ and $\mathcal{M}_c, w' \models f$. Now, by definition of $\mathcal{R}_{\mathcal{B}}$, iff $\mathcal{M}_c, a_1 \dots a_n \mathcal{B} \models f$ or $\mathcal{M}_c, a_1 \dots a_n \mathcal{M} \models f$, that is, iff (1) $V(a_1 \dots a_n \mathcal{B}, f) = \mathbf{T}$ or (2) $V(a_1 \dots a_n \mathcal{M}, f) = \mathbf{T}$. By definition of V in \mathcal{M}_c , iff (1) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$ or (2) $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$, or $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$. If $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}f$ and $a_1, \dots, a_n \vdash_\Delta \neg\mathcal{B}\neg f$ is the case, it is obvious, otherwise if $a_1, \dots, a_n \vdash_\Delta \mathcal{B}f$ is the case, we obtain the thesis by applying the rule 9.
- $G = Fs_1 \wedge Fs_2$: $\mathcal{M}_c, a_1 \dots a_n \models Fs_1 \wedge Fs_2$ iff $\mathcal{M}_c, a_1 \dots a_n \models Fs_1$ and $\mathcal{M}_c, a_1 \dots a_n \models Fs_2$; by inductive hypothesis $a_1, \dots, a_n \vdash_\Delta Fs_1$ and $a_1, \dots, a_n \vdash_\Delta Fs_2$. Hence, by definition of \vdash_Δ , rule 8, $a_1, \dots, a_n \vdash_\Delta Fs_1 \wedge Fs_2$.
- $G = \langle a \rangle G'$: $\mathcal{M}_c, a_1 \dots a_n \models \langle a \rangle G'$, where $a \in \mathcal{A}$, iff it exists $w' \in W$ s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_a$ and $\mathcal{M}_c, w' \models G'$. Now, by definition of \mathcal{R}_a in \mathcal{M}_c , we have $w' = a_1 \dots a_n a$ and $a_1, \dots, a_n \vdash_\Delta Fs'$, where $\Box(Fs' \supset \langle a \rangle \top) \in \Pi$. Moreover, from $\mathcal{M}_c, a_1 \dots a_n a \models G'$, by inductive hypothesis, $a_1, \dots, a_n, a \vdash_\Delta G'$. Hence, by definition of \vdash_Δ , rule 3, $a_1, \dots, a_n \vdash_\Delta \langle a \rangle G'$.

- $G = \langle (Fs')? \rangle G'$: $\mathcal{M}_c, a_1 \dots a_n \models \langle (Fs')? \rangle G'$ iff it exists $w' \in W$ s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{(Fs')?}$ and $\mathcal{M}_c, w' \models G'$. Now, by definition of $\mathcal{R}_{\psi?}$ in \mathcal{M}_c , we have $w' = a_1 \dots a_n$ and $a_1, \dots, a_n \vdash_{\Delta} Fs'$. Moreover, since $\mathcal{M}_c, a_1 \dots a_n \models G'$, by inductive hypothesis $a_1, \dots, a_n \vdash_{\Delta} G'$. Hence, by definition of \vdash_{Δ} , rule 3, $a_1, \dots, a_n \vdash_{\Delta} \langle (Fs')? \rangle G'$.
- $G = \langle p \rangle G'$: $\mathcal{M}_c, a_1 \dots a_n \models \langle p \rangle G'$, where $p \in \mathcal{P}$, iff it exists $w' \in W$ s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_p$ and $\mathcal{M}_c, w' \models G'$. Now, by definition of \mathcal{R}_p in \mathcal{M}_c , we have $w' = a_1 \dots a_n a_{n+1} \dots a_{n+m}$ and $p \xrightarrow{*} \Pi_{\mathcal{P}} a_{n+1} \dots a_{n+m}$. Moreover, from $\mathcal{M}_c, a_1 \dots a_n a_{n+1} \dots a_{n+m} \models G'$ by inductive hypothesis $a_1, \dots, a_n, a_{n+1}, \dots, a_{n+m} \vdash_{\Delta} G'$. From it, we proceed by iterative applications of rule 3, for dealing with atomic actions, and of rules 1 and 4 of \vdash_{Δ} , on the line of the steps of the derivation $p \Rightarrow_{\Pi_{\mathcal{P}}} p'_1 \dots p'_n \Rightarrow_{\Pi_{\mathcal{P}}} \dots \Rightarrow_{\Pi_{\mathcal{P}}} a_{n+1} \dots a_{n+m}$. At the end of the process, $a_1, \dots, a_n \vdash_{\Delta} \langle p'_1 \rangle \dots \langle p'_n \rangle G'$, and hence, by rule 1, $a_1, \dots, a_n \vdash_{\Delta} \langle p \rangle G'$.
- $G = \langle s \rangle G'$: $\mathcal{M}_c, a_1 \dots a_n \models \langle s \rangle G'$, where $s \in \mathcal{S}$, iff it exists $w' \in W$ s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_s$ and $\mathcal{M}_c, w' \models G'$. Now, by definition of \mathcal{R}_s in \mathcal{M}_c , we have $w' = a_1 \dots a_n s^{Bl}$ with $s^{Bl} \in \mathcal{A}$ and $l \in \text{dom}(s)$. Then, by inductive hypothesis $a_1, \dots, a_n, s^{Bl} \vdash_{\Delta} G'$, with $l \in \text{dom}(s)$. Hence, by definition of \vdash_{Δ} , rule 3, $a_1, \dots, a_n, \vdash_{\Delta} \langle s^{Bl} \rangle G'$, and then, by rule 4, $a_1, \dots, a_n, \vdash_{\Delta} \langle s \rangle G'$.

We prove the property (2) if we prove that for all formulae D in $\{\Pi_{\mathcal{A}} \cup \Delta \cup S_0\}$ holds that $\mathcal{M}_c, \varepsilon \models D$. We reason by cases on the structure of D .

- $D = \Box(Fs' \supset [a_i]F)$: $\mathcal{M}_c, \varepsilon \models \Box(Fs' \supset [a_i]F)$ iff for each world w' s.t. $(\varepsilon, w') \in \mathcal{R}_{\Box}$, $\mathcal{M}_c, w' \models Fs' \supset [a_i]F$. In particular, by definition of \mathcal{R}_{\Box} , w' is a generic modal context $a_1 \dots a_n$, then we have to prove $\mathcal{M}_c, a_1 \dots a_n \models Fs' \supset [a_i]F$, for each $a_1 \dots a_n$. Let us assume (a) $\mathcal{M}_c, a_1 \dots a_n \models Fs'$ and we prove the thesis (b) $\mathcal{M}_c, a_1 \dots a_n \models [a_i]F$. Note that our thesis (b) holds iff for each world w'' s.t. $(a_1 \dots a_n, w'') \in \mathcal{R}_{a_i}$, $\mathcal{M}_c, w'' \models F$, where, by definition of \mathcal{R}_{a_i} in \mathcal{M}_c , $w'' = a_1 \dots a_n a_i$. By property 1, our assumption (a) holds iff $a_1, \dots, a_n \vdash_{\Delta} Fs'$, and proving our thesis is equivalent to prove $a_1, \dots, a_n, a_i \vdash_{\Delta} F$. Since $\Box(Fs' \supset [a_i]F)$ is a clause in $\Pi_{\mathcal{A}}$, by definition of \vdash_{Δ} (rule 7a), $a_1, \dots, a_n, a_i \vdash_{\Delta} F$ follows from $a_1, \dots, a_n \vdash_{\Delta} Fs'$.
- $D = \Box(Fs' \supset \langle a_i \rangle \top)$: $\mathcal{M}_c, \varepsilon \models \Box(Fs' \supset \langle a_i \rangle \top)$ iff for each world w' s.t. $(\varepsilon, w') \in \mathcal{R}_{\Box}$, $\mathcal{M}_c, w' \models Fs' \supset \langle a_i \rangle \top$. In particular, by definition of \mathcal{R}_{\Box} in \mathcal{M}_c , w' is a generic modal context $a_1 \dots a_n$, then we have to prove $\mathcal{M}_c, a_1 \dots a_n \models Fs' \supset \langle a_i \rangle \top$, for each $a_1 \dots a_n$. Let us assume (a) $\mathcal{M}_c, a_1 \dots a_n \models Fs'$ and we prove (b) $\mathcal{M}_c, a_1 \dots a_n \models \langle a_i \rangle \top$. By property (1), our assumption (a) holds iff $a_1, \dots, a_n \vdash_{\Delta} Fs'$. Proving (b) means to prove that it exists a world w' s.t. $(a_1 \dots a_n, w') \in \mathcal{R}_{a_i}$. By definition of \mathcal{R}_{a_i} in \mathcal{M}_c , w' must be $a_1 \dots a_n a_i$ and $(a_1 \dots a_n, a_1 \dots a_n a_i) \in \mathcal{R}_{a_i}$ if $a_1, \dots, a_n \vdash_{\Delta} Fs'$, with $\Box(Fs' \supset \langle a_i \rangle \top) \in \Pi$. But $a_1, \dots, a_n \vdash_{\Delta} Fs'$ is true by our assumption (a) and $\Box(Fs' \supset \langle a_i \rangle \top) \in \Pi$ holds by hypothesis, then $(a_1 \dots a_n, a_1 \dots a_n a_i) \in \mathcal{R}_{a_i}$.
- $D = F$: F have form $\mathcal{B}f$, $\mathcal{B}\neg f$, $\neg\mathcal{B}f$, or $\neg\mathcal{B}\neg f$.

$D = \mathcal{B}f$: $\mathcal{M}_c, \varepsilon \models \mathcal{B}f$ iff for each w' s.t. $(\varepsilon, w') \in \mathcal{R}_B$, $\mathcal{M}_c, w' \models f$. Now, by definition of \mathcal{R}_B in \mathcal{M}_c , $\mathcal{M}_c, \mathcal{B} \models f$ and $\mathcal{M}_c, \mathcal{M} \models f$, that is, iff (1) $V(\mathcal{B}, f) = \mathbf{T}$ and (2) $V(\mathcal{M}, f) = \mathbf{T}$. By definition of V in \mathcal{M}_c , iff (1) $\varepsilon \vdash_{\Delta} \mathcal{B}f$ and (2) $\varepsilon \vdash_{\Delta} \mathcal{B}f$, or $\varepsilon \vdash_{\Delta} \neg \mathcal{B}f$ and $\varepsilon \vdash_{\Delta} \neg \mathcal{B}\neg f$. (1) and (2) hold because, by hypothesis, $\mathcal{B}f \in S_0$.

$D = \mathcal{B}\neg f$: $\mathcal{M}_c, \varepsilon \models \mathcal{B}\neg f$ iff for each w' s.t. $(\varepsilon, w') \in \mathcal{R}_B$, $\mathcal{M}_c, w' \models \neg f$. Now, by definition of \mathcal{R}_B in \mathcal{M}_c , $\mathcal{M}_c, \mathcal{B} \models \neg f$ and $\mathcal{M}_c, \mathcal{M} \models \neg f$, that is, iff (1) $V(\mathcal{B}, f) = \mathbf{F}$ and (2) $V(\mathcal{M}, f) = \mathbf{F}$. By definition of V in \mathcal{M}_c , iff (1) $\varepsilon \vdash_{\Delta} \mathcal{B}\neg f$, or $\varepsilon \vdash_{\Delta} \neg \mathcal{B}f$ and $\varepsilon \vdash_{\Delta} \neg \mathcal{B}\neg f$, and (2) $\varepsilon \vdash_{\Delta} \mathcal{B}\neg f$. (1) and (2) hold because, by hypothesis, $\mathcal{B}\neg f \in S_0$.

$D = \neg \mathcal{B}f$: $\mathcal{M}_c, \varepsilon \models \neg \mathcal{B}f$ iff it exists a world w' s.t. $(\varepsilon, w') \in \mathcal{R}_B$ and $\mathcal{M}_c, w' \models \neg f$. Now, by definition of \mathcal{R}_B , iff $\mathcal{M}_c, \mathcal{B} \models \neg f$ or $\mathcal{M}_c, \mathcal{M} \models \neg f$, that is, iff (1) $V(\mathcal{B}, f) = \mathbf{F}$ or (2) $V(\mathcal{M}, f) = \mathbf{F}$. By definition of V in \mathcal{M}_c , iff (1) $\varepsilon \vdash_{\Delta} \mathcal{B}\neg f$, or $\varepsilon \vdash_{\Delta} \neg \mathcal{B}f$ and $\varepsilon \vdash_{\Delta} \neg \mathcal{B}\neg f$, or (2) $\varepsilon \vdash_{\Delta} \mathcal{B}\neg f$. This disjunction holds because, by hypothesis, $\neg \mathcal{B}f \in S_0$ and, by definition 2.1, if $\neg \mathcal{B}f \in S_0$ we have that either $\mathcal{B}\neg f \in S_0$ or $\neg \mathcal{B}\neg f \in S_0$.

$G = \neg \mathcal{B}\neg f$: $\mathcal{M}_c, a\varepsilon \models \neg \mathcal{B}\neg f$ iff it exists a world w' s.t. $(\varepsilon, w') \in \mathcal{R}_B$ and $\mathcal{M}_c, w' \models f$. Now, by definition of \mathcal{R}_B , iff $\mathcal{M}_c, \varepsilon \mathcal{B} \models f$ or $\mathcal{M}_c, \varepsilon \mathcal{M} \models f$, that is, iff (1) $V(\mathcal{B}, f) = \mathbf{T}$ or (2) $V(\mathcal{M}, f) = \mathbf{T}$. By definition of V in \mathcal{M}_c , iff (1) $\varepsilon \vdash_{\Delta} \mathcal{B}f$ or (2) $\varepsilon \vdash_{\Delta} \mathcal{B}f$, or $\varepsilon \vdash_{\Delta} \neg \mathcal{B}f$ and $\varepsilon \vdash_{\Delta} \neg \mathcal{B}\neg f$. This disjunction holds because, by hypothesis, $\neg \mathcal{B}\neg f \in S_0$ and, by definition 2.1, if $\neg \mathcal{B}\neg f \in S_0$ we have that either $\neg \mathcal{B}f \in S_0$ or $\mathcal{B}f \in S_0$.

- $D = \mathbf{M}[a_1; \dots; a_m]F$: $\mathcal{M}_c, \varepsilon \models \mathbf{M}[a_1; \dots; a_m]F$ iff $V(\varepsilon, \mathbf{M}[a_1; \dots; a_m]F) = \mathbf{T}$, which by definition of V in \mathcal{M}_c holds iff $\mathbf{M}[a_1, \dots, a_m]F \in \Delta$, that is true by hypothesis. □

Now we can prove the following result.

Theorem A.3 (Completeness of \vdash_{Δ}). Let (Π, S_0) be an e-consistent dynamic domain description and let G be a query of form $\langle p_1; p_2; \dots; p_n \rangle F$ s. Let Δ be a consistent set of abductive assumptions. If $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1; \dots; a_m]G$, then there is a \vdash_{Δ} -proof Υ for the query G from the domain description (Π, S_0) at state a_1, \dots, a_m .

Proof. Our hypothesis is $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m]G$, that holds iff $\models \Sigma_{(\Pi, S_0)} \cup \Delta \supset [a_1; \dots; a_m]G$. Then, for every Kripke (Π, S_0) -interpretation and for every world $w \in W$, $M, w \models \Sigma_{(\Pi, S_0)} \cup \Delta$ implies $M, w \models [a_1; \dots; a_m]G$. Hence, in particular for the canonical model \mathcal{M}_c and the world ε , $\mathcal{M}_c, \varepsilon \models \Sigma_{(\Pi, S_0)} \cup \Delta$ implies $\mathcal{M}_c, \varepsilon \models [a_1; \dots; a_m]G$. By theorem A.2, property (2), we have that $\mathcal{M}_c, \varepsilon \models \Sigma_{(\Pi, S_0)} \cup \Delta$ holds, thus $\mathcal{M}_c, \varepsilon \models [a_1; \dots; a_m]G$ holds, and then, by theorem A.2, property (1), $a_1, \dots, a_n \vdash_{\Delta} G$. □

Now we are in the position to give the proof of the soundness of the proof procedure in section 4.1 respect to an acceptable solution.

Theorem A.4 (Soundness of \vdash_{ps}). Let (Π, S_0) be an e-consistent dynamic domain description and let $\langle p_1; p_2; \dots; p_n \rangle Fs$ be a query. For every abductive solution Δ for (Π, S_0) , for every answer σ , if $\langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds from (Π, S_0) with answer σ , then $\Sigma_{(\Pi, S_0)} \cup \Delta \models \langle p_1; p_2; \dots; p_n \rangle Fs$.

Proof. In order to prove the theorem, we prove the following property: if $a_1, \dots, a_m \vdash_{ps} \langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds (finitely fails) then $a_1, \dots, a_m \vdash_{\Delta} \langle p_1; p_2; \dots; p_n \rangle Fs$ succeeds (finitely fails). In fact, by this property and by making use of the theorems A.1 and A.3 we can conclude the thesis. The proof is by double induction on the rank r of the \vdash_{ps} -proof Υ of the query $\langle p_1; p_2; \dots; p_n \rangle Fs$, that is on its nesting level, starting from the innermost one, and the height h of Υ .

Let Υ be a \vdash_{ps} -proof of rank $r = 0$ and height $h = 1$, then Υ is an axiom and the property holds trivially.

By inductive hypothesis the theorem holds for queries whose proof Υ has height less than or equal to h . Let us prove it for $h + 1$.

Let us consider the case that the query succeeds. We consider the following cases, one for each inference rule in which Υ can terminate.

Case rule 1. Assume that the root inference figure in Υ is rule 1. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon' \quad a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{ps} \langle p; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $p \in \mathcal{P}$ and $\langle p \rangle \varphi \subset \langle p'_1; \dots; p'_n \rangle \varphi \in \Pi_{\mathcal{P}}$. Since Υ' is shorter than Υ , by inductive hypothesis, we get that if $a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs$ succeeds then $a_1, \dots, a_m \vdash_{\Delta} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs$ succeeds. Now, by application of rule 1 of \vdash_{Δ} we obtain the thesis.

Cases rule 2, rule 3, rule 4, rule 5, rule 7a, rule 8, rule 9 are similar.

Case rule 7c. Trivial, because the case never arises when the rank r is 0.

Let us consider the case the query finitely fails. We consider the following cases, one for each inference rule in which an attempt Υ_i to prove the query can terminate.

Case rule 1. Assume that the root inference figure in a Υ_i is rule 1. Hence, in our hypothesis, Υ_i has form:

$$\frac{\text{every attempt } \Upsilon'_i \text{ finitely fails} \quad a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{ps} \langle p; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $p \in \mathcal{P}$ and $\langle p \rangle \varphi \subset \langle p'_1; \dots; p'_n \rangle \varphi \in \Pi_{\mathcal{P}}$. Since every Υ'_i is shorter than Υ_i , by inductive hypothesis, we get that if $a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs$ finitely

fails then $a_1, \dots, a_m \vdash_{\Delta} \langle p; p_2; \dots; p_n \rangle Fs$ finitely fails. Then, the query finitely fails too.

Cases rule 2, rule 3, rule 4, rule 5, rule 7a, rule 8, rule 9 are similar.

Case rule 7c. Trivial, because the case never arises when the rank r is 0.

Let Υ be a \vdash_{ps} -proof of rank $r + 1$ and height $h + 1$ (the case of rank $r + 1$ and height $h = 1$ is trivially true because it never arises). By inductive hypothesis the theorem holds for queries whose proof Υ has rank less or equal r and height less than or equal to h . Let us prove it for the rank $r + 1$ and height $h + 1$.

Let us consider the case the query succeeds. We consider the following cases, one for each inference rule in which Υ can terminate.

Case rule 1. Assume that the root inference figure in Υ is rule 1. Hence, in our hypothesis, Υ has form:

$$\frac{\Upsilon' \quad a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{ps} \langle p; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $p \in \mathcal{P}$ and $\langle p \rangle \varphi \subset \langle p'_1; \dots; p'_n \rangle \varphi \in \Pi_{\mathcal{P}}$. Since Υ' is shorter than Υ , by inductive hypothesis, we get that if $a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs$ succeeds then $a_1, \dots, a_m \vdash_{\Delta} \langle p'_1; \dots; p'_n; p_2; \dots; p_n \rangle Fs$ succeeds. Now, by application of rule 1 of \vdash_{Δ} we obtain the thesis.

Cases rule 2, rule 3, rule 4, rule 5, rule 7a, rule 8, rule 9 are similar.

Case rule 7c. Assume that the root inference figure in Υ is rule 1. If $a_1, \dots, a_m \vdash_{fs} F$ succeeds, then Υ has form:

$$\frac{\text{every attempt } \Upsilon_i \text{ finitely fails} \quad \Upsilon'' \quad \mathbf{not} \ a_1, \dots, a_m \vdash_{fs} \neg F \quad a_1, \dots, a_{m-1} \vdash_{fs} F}{a_1, \dots, a_m \vdash_{fs} F}$$

where $m > 0$. Since Υ'' is shorter than Υ , by inductive hypothesis, we get that if $a_1, \dots, a_{m-1} \vdash_{fs} F$ succeeds then $a_1, \dots, a_{m-1} \vdash_{\Delta} F$ succeeds. Moreover, if $a_1, \dots, a_m \vdash_{fs} F$ succeeded, then every possible \vdash_{fs} -proof Υ_i (of rank less or equals to m) of $a_1, \dots, a_m \vdash_{fs} \neg F$ must finitely fail. From this, by inductive hypothesis we have that $a_1, \dots, a_m \vdash_{\Delta} \neg F$ finitely fails and for theorem A.3 (contraposition) $\Sigma_{(\Pi, S_0)} \cup \Delta \not\models [a_1; \dots; a_m] \neg F$. Since Δ is an abductive solution, by definition 3.2 (maximality condition) $\mathbf{M}[a_1; \dots; a_m] F \in \Delta$. Then, by definition A.1, we have that $a_1, \dots, a_m \vdash_{\Delta} F$ can be derived by rule 7c'.

Let us consider the case the query finitely fails. We consider the following cases, one for each inference rule in which all attempt Υ_i to prove the query can terminate.

Case rule 1. Assume that the root inference figure in Υ_i is rule 1. Hence, in our hypothesis, Υ_i has form:

$$\frac{\text{every attempt } \Upsilon'_i \text{ finitely fails} \quad a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_{n'}; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}{a_1, \dots, a_m \vdash_{ps} \langle p; p_2; \dots; p_n \rangle Fs \text{ w.a. } \sigma}$$

where $p \in \mathcal{P}$ and $\langle p \rangle \varphi \subset \langle p'_1; \dots; p'_{n'} \rangle \varphi \in \Pi_{\mathcal{P}}$. Since every Υ'_i is shorter than Υ_i , by inductive hypothesis, we get that if $a_1, \dots, a_m \vdash_{ps} \langle p'_1; \dots; p'_{n'}; p_2; \dots; p_n \rangle Fs$ finitely fails then $a_1, \dots, a_m \vdash_{\Delta} \langle p'_1; \dots; p'_{n'}; p_2; \dots; p_n \rangle Fs$ finitely fails. Then the query finitely fails too.

Cases rule 2, rule 3, rule 4, rule 5, rule 7a, rule 8, rule 9 are similar.

Case rule 7c. Let us consider the case $a_1, \dots, a_m \vdash_{fs} F$ and every possible \vdash_{fs} -proof Υ_i has form:

$$\frac{\dots \quad \Upsilon''_i \quad \mathbf{not} \ a_1, \dots, a_m \vdash_{fs} \neg F \quad a_1, \dots, a_{m-1} \vdash_{fs} F}{a_1, \dots, a_m \vdash_{fs} F}$$

with $m > 0$ and where or $\mathbf{not} \ a_1, \dots, a_m \vdash_{fs} \neg F$ finitely fails or $a_1, \dots, a_{m-1} \vdash_{fs} F$ finitely fails. The case $a_1, \dots, a_{m-1} \vdash_{fs} F$ finitely fails is a simple application of induction hypothesis on the height of the proof (same rank). If $\mathbf{not} \ a_1, \dots, a_m \vdash_{fs} \neg F$ finitely fails, then there exists a \vdash_{fs} -proof Υ'_i (of rank less or equals to r) of $a_1, \dots, a_m \vdash_{fs} \neg F$ that succeeds. From this, by inductive hypothesis we have that $a_1, \dots, a_m \vdash_{\Delta} \neg F$ succeeds and for theorem A.3 (contraposition) $\Sigma_{(\Pi, S_0)} \cup \Delta \models [a_1, \dots, a_m] \neg F$. Then, since Δ is an abductive solution, by definition 3.2 $\mathbf{M}[a_1; \dots; a_m] F \notin \Delta$. Then, by definition A.1, we have that $a_1, \dots, a_m \vdash_{\Delta} F$ finitely fails. \square

References

- [1] ALiCE, Advanced Logic in Computing Environments, Dipartimento di Informatica, Università degli Studi di Torino (2001). Web site: <http://www.di.unito.it/~alice>.
- [2] M. Baldoni, Normal multimodal logics: Automatic deduction and logic programming extension, PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy (1998). Available at <http://www.di.unito.it/~baldoni/>.
- [3] M. Baldoni, Normal multimodal logics with interaction axioms, in: *Labelled Deduction*, eds. D. Basin, M. D'Agostino, D.M. Gabbay, S. Matthews and L. Viganò, Applied Logic Series, Vol. 17 (Kluwer Academic, 2000) pp. 33–53.
- [4] M. Baldoni, C. Baroglio, A. Chiarotto and V. Patti, Programming goal-driven web sites using an agent logic language, in: *Proc. of the Third International Symposium on Practical Aspects of Declarative Languages*, ed. I.V. Ramakrishnan, Lecture Notes in Computer Science, Vol. 1990, Las Vegas, NV (Springer, 2001) pp. 60–75.
- [5] M. Baldoni, C. Baroglio, L. Giordano, A. Martelli and V. Patti, Reasoning about communicating agents in the semantic web, in: *Proc. of the 1st International Workshop on Principle and Practice of Semantic Web Reasoning (PPSWR'2003), ICLP 2003*, Lecture Notes in Computer Science, Mumbai, India, 2003 (Springer, to appear).

- [6] M. Baldoni, C. Baroglio, A. Martelli and V. Patti, Reasoning about conversation protocols in a logic-based agent language, in: *AI*IA 2003: Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence*, eds. A. Cappelli and F. Turini, Lecture Notes in Computer Science, Vol. 2829, Pisa, Italy (Springer, 2003) pp. 300–311.
- [7] M. Baldoni, C. Baroglio, A. Martelli and V. Patti, Reasoning about self and others: communicating agents in a modal action logic, in: *Proc. of Theoretical Computer Science, 8th Italian Conference, ICTCS'2003*, eds. C. Blundo and C. Laneve, Lecture Notes in Computer Science, Vol. 2841, Bologna, Italy (Springer, 2003) pp. 228–241.
- [8] M. Baldoni, C. Baroglio and V. Patti, Structureless, intention-guided web sites: Planning based adaptation, in: *Proc. of the 9th International Conference on Human-Computer Interaction (HCII 2001), Symposium on Human Interfaces 2001, 4th International Conference on Engineering Psychology and Cognitive Ergonomics, 1th International Conference on Universal Access in Human-Computer Interaction*, ed. C. Stephanidis, Vol. 3, New Orleans, LA (Lawrence Erlbaum Associates, 2001) pp. 237–241.
- [9] M. Baldoni, C. Baroglio and V. Patti, Applying logic inference techniques for gaining flexibility and adaptivity in tutoring systems, in: *Proc. of the 10th International Conference on Human-Computer Interaction (HCII 2003), Symposium on Human Interfaces 2003, 5th International Conference on Engineering Psychology and Cognitive Ergonomics, 2th International Conference in Human-Computer Interaction*, ed. C. Stephanidis, Vol. 4, Crete, Greece (Lawrence Erlbaum Associates, 2003) pp. 517–521.
- [10] M. Baldoni, C. Baroglio, V. Patti and L. Torasso, Using a rational agent in an adaptive web-based tutoring system, in: *Proc. of Workshop on Adaptive System for Web-based Education, 2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems (AH 2002), Selected Papers*, eds. P. Brusilovsky, N. Henze and E. Millan, Malaga, Spain (2002) pp. 43–55.
- [11] M. Baldoni, L. Giordano and A. Martelli, A tableau calculus for multimodal logics and some (un)decidability results, in: *Proc. of TABLEAUX'98*, ed. H. de Swart, Lecture Notes in Artificial Intelligence, Vol. 1397 (Springer, 1998) pp. 44–59.
- [12] M. Baldoni, L. Giordano, A. Martelli and V. Patti, An abductive proof procedure for reasoning about actions in modal logic programming, in: *Proc. of NMELP'96*, eds. J. Dix et al., Lecture Notes in Artificial Intelligence, Vol. 1216 (Springer, 1997) pp. 132–150.
- [13] C. Baral and T.C. Son, Approximate reasoning about actions in presence of sensing and incomplete information, in: *Proc. of ILPS'97*, ed. J. Małuszynski (MIT Press, Cambridge, MA, 1997) pp. 387–404.
- [14] C. Baral and T.C. Son, Formalizing sensing actions – A transition function based approach, *Artificial Intelligence* 125(1–2) (2001) 19–91.
- [15] P. Brusilovsky, Adaptive hypermedia, *User Modeling and User-Adapted Interaction* 11 (2001) 87–110.
- [16] M. Castilho, O. Gasquet and A. Herzig, Modal tableaux for reasoning about actions and plans, in: *Proc. ECP'97*, ed. S. Steel, Lecture Notes in Artificial Intelligence (1997) pp. 119–130.
- [17] G. De Giacomo, L. Iocchi, Daniele Nardi and Riccardo Rosati, Moving a robot: The KR & R approach at work, in: *Proc. of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, eds. L.C. Aiello, J. Doyle and S.C. Shapiro (Morgan Kaufmann, Cambridge, MA, 1996) pp. 198–209.
- [18] G. De Giacomo, L. Iocchi, D. Nardi and R. Rosati, Planning with sensing for a mobile robot, in: *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97*, eds. S. Steel and R. Alami, Lecture Notes in Computer Science, Vol. 1348, Toulouse, France (Springer, 1997) pp. 156–168.
- [19] G. De Giacomo and M. Lenzerini, PDL-based framework for reasoning about actions, in: *Proc. of AI*IA '95*, Lecture Notes in Artificial Intelligence, Vol. 992 (1995) pp. 103–114.
- [20] G. De Giacomo, Y. Lespérance and H.J. Levesque, Reasoning about concurrent execution, prioritized

- interrupts, and exogenous actions in the situation calculus, in: *Proc. of IJCAI'97*, Nagoya (August 1997) pp. 1221–1226.
- [21] G. De Giacomo and H.J. Levesque, An incremental interpreter for high-level programs with sensing, in: *Proc. of the AAAI 1998 Fall Symposium on Cognitive Robotics*, Orlando, FL (October 1998).
- [22] G. De Giacomo and R. Rosati, Minimal knowledge approach to reasoning about actions and sensing, in: *Proc. of the IJCAI'99 Workshop on Nonmonotonic Reasoning, Action and Change (NRAC'99)*, Stockholm, Sweden (August 1999) pp. 25–31.
- [23] M. Denecker and D. De Schreye, Representing incomplete knowledge in abduction logic programming, in: *Proc. of ILPS '93* (MIT Press, Vancouver, 1993).
- [24] F. Dignum and M. Greaves, Issues in agent communication, in: *Issues in Agent Communication*, eds. F. Dignum and M. Greaves, Lecture Notes in Computer Science, Vol. 1916 (Springer, 2000) pp. 1–16.
- [25] K. Eshghi and R. Kowalski, Abduction compared with negation by failure, in: *Proc. of ICLP '89* (MIT Press, Lisbon, 1989).
- [26] FIPA, Fipa 97, specification part 2: Agent communication language. Technical report, FIPA (Foundation for Intelligent Physical Agents) (November 1997). Available at: <http://www.fipa.org/>.
- [27] M. Gelfond and V. Lifschitz, Representing action and change by logic programs, *Journal of Logic Programming* 17 (1993) 301–321.
- [28] G. De Giacomo, Y. Lespérance, H.J. Levesque and S. Sardina, On the semantic of deliberation in Indigolog: from theory to implementation, in: *Proc. of KR 2002* (2002) pp. 603–614.
- [29] L. Giordano, A. Martelli and C. Schwind, Ramification and causality in a modal action logic, *Journal of Logic and Computation* 10(5) (2000) 625–662.
- [30] D. Harel, Dynamic logic, in: *Handbook of Philosophical Logic*, eds. D. Gabbay and F. Guenther, Vol. II (D. Reidel, 1984) pp. 497–604.
- [31] S. Hölldobler and H.P. Störr, Reasoning about complex actions, in: *Proc. of NMR'98 – Action and Causality*, ed. V. Lifschitz (1998) pp. 1–9.
- [32] G.E. Hughes and M.J. Cresswell, *A New Introduction to Modal Logic* (Routledge, 1968).
- [33] H.J. Levesque, What is planning in the presence of sensing? in: *Proc. of the AAAI-96* (1996) pp. 1139–1146.
- [34] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin and R.B. Scherl, GOLOG: A logic programming language for dynamic domains, *J. of Logic Programming* 31 (1997) 59–83.
- [35] J. Lobo, G. Mendez and S.R. Taylor, Adding knowledge to the action description language \mathcal{A} , in: *Proc. of AAAI'97/IAAI'97*, Menlo Park (1997) pp. 454–459.
- [36] V. Patti, Programming rational agents: A modal approach in a logic programming setting, PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy (2002). Available at <http://www.di.unito.it/~patti/>.
- [37] R.P.A. Petrick and F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: *6th AI Planning and Scheduling Conference, AIPS 2002*, Toulouse (2002).
- [38] H. Prendinger and G. Schurz, Reasoning about action and change. A dynamic logic approach, *Journal of Logic, Language, and Information* 5(2) (1996) 209–245.
- [39] F. Sadri, F. Toni and P. Torroni, Dialogues for negotiation: Agent varieties and dialogue sequences, in: *Proc. of ATAL'01*, Seattle, WA (2001).
- [40] R. Scherl and H.J. Levesque, The frame problem and knowledge-producing actions, in: *Proc. of the AAAI-93*, Washington, DC (1993) pp. 689–695.
- [41] C.B. Schwind, A logic based framework for action theories, in: *Language, Logic and Computation*, eds. J. Ginzburg et al. (CSLI, 1997) pp. 275–291.
- [42] M. Thielscher, Representing the knowledge of a robot, in: *Proc. of the International Conference on Principles of Knowledge Representation and Reasoning, KR'00* (Morgan Kaufmann, 2000) pp. 109–120.