Electronic Notes in Theoretical Computer Science 48 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume48.html pp. 1 – 26

Probabilistic behaviours of reactive agents

Antonio Brogi [1]

Dipartimento di Informatica
Università di Pisa
Pisa, Italy

Abstract

We present a simple logic-based formalisation of the behaviours of agents capable of reacting to changes occurring in the external environment. Logic programming is chosen as the specification language of agents, and a quantitative analysis of the behaviours of reactive agents is described.

1 Introduction

Rationality and reactivity are two capabilities of primary importance in multi-agent systems. Agents must be able to exhibit a rational behaviour as well as to promptly and adequately react to changes occurring in the external environment.

One of the problems is that while developing a reactive agent, the environment in which the agent will operate is at least partially unknown. Typically, even if the set of possible observable behaviours of the environment is known, the precise dynamic behaviour of the environment is not predictable at software development time. On the other hand, the availability of a well-founded description of the possible behaviours of a reactive program is crucial for performing tasks such as verification and analysis before putting the program at work with the external environment.

In this paper, we will describe a simple logic-based formalisation of the behaviours of agents that feature a combination of rational and reactive capabilities. We choose logic programming as the specification language of agents. Logic programming supports a declarative, high-level programming style (from algorithmic programming to databases to artificial intelligence applications) via a small number of powerful features: Nondeterminism, unification, and recursion. Moreover, because of their operational interpretation, logic programs can be viewed as high-level, executable specifications of reactive programs.

[1] Email: brogi@di.unipi.it

We first show, following [4], how the possible behaviours of an agent reacting to different environments at different times can be modeled by means of a mapping over sets of Herbrand interpretations. The obtained formal characterisation of the possible behaviours of reactive programs provides a firm basis for reasoning about them.

For instance, one of the properties of interest in the practice of multi-agent systems is whether a reactive agent may be able to possibly derive certain conclusions depending on the evolution of the external environment. The dual property is even more important for many applications, especially for critical ones. Namely the agent behaviour should satisfy certain mandatory requirements independently of the way in which the external environment will evolve. The notions of *possible beliefs* and *invariant* of a program were introduced in [4] to formally characterise such properties, and different ways of combining rationality and reactivity were compared one another on that basis.

The aim of this paper is to introduce quantitative aspects in the analysis of the possible behaviours of reactive agents. We start from the observation that, while the evolution of the external environment is not known *a priori*, some environment perceptions are indeed more probable than others. The above observation suggests that taking into account the probabilities associated with environment perceptions paves the way for a *probabilistic* — rather than *possibilistic* — analysis of the behaviours of reactive agents.

We show how the formal setting introduced in [4] can be smoothly extended to model probabilistic behaviours of agents. We then introduce the notions of *probabilistic beliefs* and *probabilistic invariant* that can be exploited to perform effective, resource bounded analyses of the probabilistic behaviours of reactive agents.

2 Background

We will use standard notions and terminology of logic programming [3]. Since we will focus on bottom-up computations of *definite* logic programs, which are formally defined via the so-called *immediate consequence operator* $T(P)$, we recall here the definition of that operator.

Definition 2.1 Given a logic program P and a Herbrand interpretation I, the immediate consequence operator $T(P)$ is the mapping defined as follows:

$$T(P)(I) = \{A \mid \exists \overline{B} : \ A \leftarrow \overline{B} \in \mathit{ground}(P) \ \wedge \ \overline{B} \subseteq I\}$$

where $\overline{B}$ is a (possibly empty) set of atoms, and where $\mathit{ground}(P)$ denotes the set of ground instances of clauses of program P. □

The set $T(P)(I)$ denotes the set of conclusions that a program P is able to derive, in a single computation step, from a set of premises I which are assumed to hold. The powers of $T(P)$ are defined in the standard way:

$$T^0(P)(I) = I$$
$$T^{n+1}(P)(I) = T(P)\ (T^n(P)(I))$$
$$T^\omega(P)(I) = \bigcup_{i \in \omega} T^i(P)(I)$$

$T(P)$ is a continuous mapping from (Herbrand) interpretations to (Herbrand) interpretations and the power set $\mathcal{P}(\mathcal{B})$ of the Herbrand base $\mathcal{B}$ is a complete lattice under set inclusion. The immediate consequence operator has therefore a least fixpoint which coincides with the union of the finite powers of $T(P)$ applied to the bottom element $\emptyset$, namely: $lfp(T(P)) = T^\omega(P)(\emptyset)$.

3 Modeling reactive programs

3.1 Environment representation

Agents have a partial representation of the external environment and limited capabilities of interacting with it. An agent typically represents the external environment in terms of its individual *perceptions* of the environment. The type of such perceptions of course depends on the sensing capabilities owned by the agent.

We will focus on the way in which the behaviour of an agent may be influenced by its perceptions of the external environment, rather than on the way in which the agent will get such perceptions. For instance, we will abstract from the way in which a software agent accesses some piece of information available in the external environment (e.g., by receiving a message, by downloading a file, or by getting data from physical sensors).

At each moment, the external environment is hence represented by the perceptions of the agent. When agents are specified by logic programs, environment perceptions can be naturally represented as Herbrand interpretations. Moreover, while an agent is performing its computation, the external environment may arbitrarily and independently evolve. We therefore define an environment representation as follows.

Definition 3.1 An *environment representation* is a (possibly infinite) family

$$\mathcal{E} = \{E_1, E_2, E_3, \ldots\}$$

of *perceptions*, where each perception E_i is a Herbrand interpretation. □

Intuitively speaking, a Herbrand interpretation E_i represents the perception of the external environment that the agent may have at a given moment. Logically speaking, E_i represents a set of formulae that a program P may assume to be true when it performs a computation step by reacting to the current environment perceived as E_i.

The assumption that the environment may arbitrarily change during the program computation is mirrored by the flat structure of an environment

representation, which is a set of possibly non-related interpretations. (Namely there may exist pairs of environment perceptions which are not comparable one another under set-theoretic inclusion.)

3.2 Program behaviours

As shown in [4], the behaviour of a reactive program can be formally defined by means of a continuous mapping $\mathcal{T}_\varphi$ over sets of interpretations, whose least fixpoint characterises the set of all possible behaviours of a program reacting to different environments at different times.

The formal definition of the mapping $\mathcal{T}_\varphi$ is given by extending the standard immediate consequence operator $T(P)$ in two steps:

(1) First to take into account the external environment, and

(2) then to consider *sets* of interpretations.

Let us first consider how environment perceptions can be taken into account in the formalisation of the bottom-up behaviour of a program. At each computation step, a program may react to the environment, that is, to one of the environment perceptions in the environment representation $\mathcal{E}$. We correspondingly introduce the set of *extended* immediate consequences $\varphi(P)(I, E)$ of a program P, starting from a set of atoms I assumed to be true and reacting to an environment perception E. Namely, for each program P, the mapping $\varphi(P)$ given a pair of interpretations yields a single interpretation:

$$\varphi(P) : \ \mathcal{P}(\mathcal{B}) \ \times \ \mathcal{P}(\mathcal{B}) \rightarrow \mathcal{P}(\mathcal{B}).$$

The mapping $\varphi(P)$ defines the way in which a program P takes into account an environment perception when performing a deduction step. Different ways of defining $\varphi(P)$ are presented and compared in [4]. We will introduce here the simplest, and perhaps most natural, definition of $\varphi(P)$.

Definition 3.2 Let P be a program. For each Herbrand interpretation I and each environment perception E, we put:

$$\varphi(P)(I, E) \ = \ T(P)(I \cup E).$$

□

Namely the set $\varphi(P)(I, E)$ is the set of consequences that P may draw in one deduction step by assuming the set of formulae $(I \cup E)$ to be true.

Example 3.3 Consider for instance a walking robot that controls its walking speed on the base of its perception of the weather. Suppose that the robot can move at two different speeds (slow or fast), and that it is able to perceive three weather conditions (sun, rain and snow). A natural specification of the speed control is to move slowly when it rains, to move fast when it is sunny, and not to move when it is snows. The speed control of such a robot can then be described by the following program P:

```
Slow  <- Still, Sun
Slow  <- Still, Rain
Still <- Still, Snow

Fast  <- Slow, Sun
Slow  <- Slow, Rain
Still <- Slow, Snow

Fast  <- Fast, Sun
Slow  <- Fast, Rain
Still <- Fast, Snow
```

where the set of environment perceptions is $\mathcal{E}$ = { {Sun},{Rain},{Snow} }.

If the robot is still then the effect of the environment perception {Rain} is:

$$\varphi(P)(\{\texttt{Still}\}, \{\texttt{Rain}\}) = \{\texttt{Slow}\}.$$

If we now consider the situation in which the robot is walking slowly and the environment perception is {Sun} we have:

$$\varphi(P)(\{\texttt{Slow}\}, \{\texttt{Sun}\}) = \{\texttt{Fast}\}.$$

□

The second extension to the immediate consequence operator consists of moving from interpretations to sets of interpretations. Indeed we want to model all possible computations of a program that may react to different environment perceptions at different steps. To this end, we introduce an operator $\mathcal{U}_\varphi$ over sets of Herbrand interpretations whose intuitive meaning is to collect the set of all possible (one-step) reactive deductions of a program starting from a set of possible interpretations and from a set of possible environment perceptions.

Definition 3.4 Given a program P, a set of Herbrand interpretations $\mathcal{I}$, and an environment representation $\mathcal{E}$, the mapping $\mathcal{U}_\varphi$ is defined as follows:

$$\mathcal{U}_\varphi(P, \mathcal{E})(\mathcal{I}) \;=\; \{\; J \mid \exists E \in \mathcal{E}, I \in \mathcal{I} :\; J = \varphi(P)(I, E) \;\}.$$

□

Intuitively speaking, the set of interpretations $\mathcal{U}_\varphi(P, \mathcal{E})(\mathcal{I})$ denotes the set of all possible one-step "φ-evolutions" of P starting from the set $\mathcal{I}$ of hypotheses and reacting to one of the environment perceptions in $\mathcal{E}$.

Notice that the definition of $\mathcal{U}_\varphi$ is parametric w.r.t. the object mapping φ over Herbrand interpretations. If we unfold the above definition of $\mathcal{U}_\varphi$ by using the formulation of φ given in Definition 3.2, we obtain:

$$\mathcal{U}_\varphi(P, \mathcal{E})(\mathcal{I}) = \{\; J \mid \exists E \in \mathcal{E}, I \in \mathcal{I} :\; J = T(P)(I \cup E) \;\}.$$

Example 3.5 *Consider again the simple program P and the environment representation $\mathcal{E}$ = { {*`Sun`*},{*`Rain`*},{*`Snow`*} } of example 3.3. We have that:*

$\mathcal{U}_\varphi(P,\mathcal{E})(\{\{\texttt{Still}\}\}) \;=\; \{\{\texttt{Still}\},\{\texttt{Slow}\}\}$

and

$\mathcal{U}_\varphi(P,\mathcal{E})(\{\{\texttt{Still}\},\{\texttt{Slow}\}\}) \;=\; \{\{\texttt{Still}\},\{\texttt{Slow}\},\{\texttt{Fast}\}\}.$ □

An inflationary version $\mathcal{T}_\varphi$ of the $\mathcal{U}_\varphi$ operator was introduced in [4] in order to obtain a fixpoint characterisation of the set of all possible reactive behaviours of a program. Namely:

$$\mathcal{T}_\varphi(P,\mathcal{E})(\mathcal{I}) \;=\; \mathcal{I} \cup \mathcal{U}_\varphi(P,\mathcal{E})(\mathcal{I})$$

The operator $\mathcal{T}_\varphi(P,\mathcal{E})$ is defined in [4] as a mapping over sets of interpretations, whose domain $\mathcal{D}$ is the power set of the power set of $\mathcal{B}$ without least element [2]. As shown in [4], $(\mathcal{D},\subseteq)$ is a cpo without bottom element, the mapping $\mathcal{T}_\varphi(P,\mathcal{E})$ is continuous on $(\mathcal{D},\subseteq)$, for each mapping φ, and therefore $\mathcal{T}_\varphi(P,\mathcal{E})$ has a least fixpoint.

The powers of ($\mathcal{U}_\varphi$ and) $\mathcal{T}_\varphi$ are defined in the standard way:

$$\begin{aligned}
\mathcal{T}_\varphi^0(P,\mathcal{E})(\mathcal{I}) &= \mathcal{I}\\
\mathcal{T}_\varphi^{n+1}(P,\mathcal{E})(\mathcal{I}) &= \mathcal{T}_\varphi(P,\mathcal{E})\ (\mathcal{T}_\varphi^{n}(P,\mathcal{E})(\mathcal{I}))\\
\mathcal{T}_\varphi^{\omega}(P,\mathcal{E})(\mathcal{I}) &= \bigcup_{i\in\omega} \mathcal{T}_\varphi^{i}(P,\mathcal{E})(\mathcal{I})
\end{aligned}$$

Moreover, as shown in [4], the least fixpoint of $\mathcal{T}_\varphi(P,\mathcal{E})$ can be computed by repeatedly applying $\mathcal{T}_\varphi(P,\mathcal{E})$ to some initial element of the domain $\mathcal{D}$. Namely, for each P, $\mathcal{E}$ and $\mathcal{I}$:

$$\bigcup_{i\in\omega} \mathcal{T}_\varphi^{i}(P,\mathcal{E})(\mathcal{I}) \;=\; min\{\mathcal{J} \mid \mathcal{J} = \mathcal{T}_\varphi(P,\mathcal{E})(\mathcal{J}) \wedge \mathcal{I} \subseteq \mathcal{J}\}.$$

That is, for each initial set $\mathcal{I}_0$ of interpretations, the least fixpoint of $\mathcal{T}_\varphi(P,\mathcal{E})$ greater than or equal to $\mathcal{I}_0$ can be computed by repeatedly applying $\mathcal{T}_\varphi(P,\mathcal{E})$ starting from the initial set of interpretations $\mathcal{I}_0$.

As illustrated in [4], the information contained in the least fixpoint of $\mathcal{T}_\varphi(P,\mathcal{E})(\mathcal{I})$ can be analysed by viewing the least fixpoint as a directed graph whose nodes are the interpretations in the least fixpoint and where labelled edges denote relations between interpretations. More precisely, the graph contains a directed edge labelled E from node I to node J if and only if $J = \varphi(P)(I,E)$. Namely an edge from I to J indicates that program P may derive the set J of consequences by means of a φ step starting from

[2] The least element of $\mathcal{P}(\mathcal{P}(\mathcal{B}))$ (viz., the empty set of interpretations) is not included in the domain since it is not a sensible argument for the $\mathcal{T}_\varphi$ operator. Indeed, intuitively speaking, the situation "nothing is assumed to be true in the environment" corresponds to the singleton set $\{\emptyset\}$ containing only the empty interpretation, rather than to the empty set $\{\}$ of interpretations.

I and reacting to the environment perception E. For instance, the graph representation of the least fixpoint of $\mathcal{T}_\varphi(P, \mathcal{E})(\{\texttt{Still}\})$ for program P of example 3.3 is illustrated in figure 1.

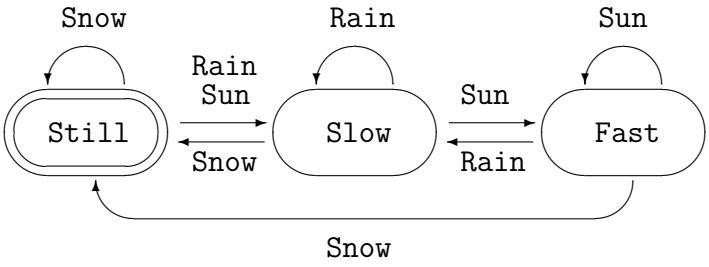

Fig. 1. Possible behaviours of the walking robot of example 3.3

4 Probabilistic behaviours of reactive agents

4.1 Motivating example

Consider an agent gambling at a roulette table. Suppose that the strategy of the agent simply consists of gambling one dollar on the red colour at each turn. Since colour bets are paid 1:1, the agent will either win or loose one dollar for each bet.

The behaviour of such an agent is specified by the following program P:

```
Holds(S(S(x))) <- Holds(S(x)), Red
Holds(x)       <- Holds(S(x)), Black
Holds(Zero)    <- Holds(Zero)
```

where the set of possible environment perceptions is $\mathcal{E} = \{$ {Red},{Black} $\}$.

Suppose that the initial budget of the agent is 100$. After the first round the agent will have either 99$ or 101$, depending on the result of wheel revolvement. After the second round the agent will have 98$, 100$, or 102$, and so on and so forth. The possible behaviours of the agent are illustrated by the graph of Figure 2 denoting the least fixpoint of $\mathcal{T}_\varphi(P, \mathcal{E})(\{\{\texttt{Holds(100)}\}\})$.

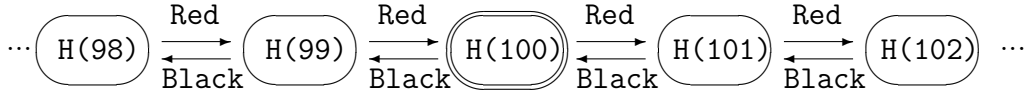

Fig. 2. Possible behaviours of the roulette player

This example highlights that while the mapping $\mathcal{T}_\varphi$ supports the analysis of the *possible* behaviours of a reactive agent, it does not give any information on the *probabilistic* behaviours of the agent.

In order to perform a *probabilistic* analysis of the behaviours of P, we can make some assumptions on the probabilities of the environment perceptions.

Formally, this corresponds to introducing a probability distribution δ on $\mathcal{E}$, that is, a total mapping δ from $\mathcal{E}$ to [0,1] such that $\sum_{E \in \mathcal{E}} \delta(E) = 1$.

The obvious values of δ for this example[3] are

$$\delta(\{\texttt{Red}\}) = .5 \text{ and } \delta(\{\texttt{Black}\}) = .5.$$

We observe that, according to the above distribution δ, after the first round the agent will have 99$ with probability .5 and 101$ with probability .5. After the second round, the agent will have 98$ with probability .25, 102$ with probability .25, and the initial amount of 100$ with probability .5, and so on and so forth, as illustrated by the following table:

round	...	96$	97$	98$	99$	100$	101$	102$	103$	104$	...
0						1					
1					.5		.5				
2				.25		.5		.25			
3			.125		.375		.375		.125		
4		.0625		.25		.375		.25		.0625	

The example suggests that taking onto account the probabilities induced by the environment perceptions may pave the way for a probabilistic analysis of agent behaviours.

4.2 From interpretations to probabilistic interpretations

We now show how the setting introduced in Section 3 can be smoothly extended in order to account for a probabilistic (rather than possibilistic) analysis of the behaviours of a reactive agent.

Intuitively speaking, we will associate probabilities to interpretations and we will consider mappings over (sets of) pairs:

$$\langle \text{interpretation, associated-probability} \rangle.$$

As a first step, we introduce the notion of *probabilistic interpretation*. A probabilistic interpretation is simply a Herbrand interpretation with a probability associated with it.

Definition 4.1 A *probabilistic interpretation* is a pair $\langle I, p \rangle$ where $I \subseteq \mathcal{B}$ and $p \in (0, 1]$. □

[3] For the sake of simplicity, we do not consider here the presence of "zero" on the wheel. Modern wheels contain two "zero" so that the probability of a red number is $\frac{16}{38}$ rather than $\frac{16}{36}$, but this is not relevant in the scope of our discussion.

In the previous section, we have informally discussed the idea of associating a probability distribution with the set of environment perceptions of an agent. Following Definition 4.1, a probabilistic environment representation can be now formally defined as a set of probabilistic interpretations. We will consider sets of probabilistic interpretations such that the probabilities associated with the interpretations form a probability distribution, as stated by the following definition.

Definition 4.2 A set $\mathcal{S}_\pi$ of probabilistic interpretations is *complete* if and only if $\sum_{\langle I,p\rangle \in \mathcal{S}_\pi} p = 1$. □

Definition 4.3 A *probabilistic environment representation*

$$\mathcal{E}_\pi = \{\langle E_1, p_1\rangle, \langle E_2, p_2\rangle, \langle E_3, p_3\rangle, \ldots\}$$

is a complete set of probabilistic interpretations. □

For instance the probabilistic environment representation for the example discussed in section 4.1 is: $\mathcal{E}_\pi = \{\langle\{\texttt{Red}\}, .5\rangle, \langle\{\texttt{Black}\}, .5\rangle\}$.

In Section 3 we introduced the mapping $\varphi(P)$ to model the way in which a program P reacts to a program perception E given a set of hypotheses I. Formally, the mapping $\varphi(P)$ given a pair of interpretations returns a single interpretation:

$$\varphi(P) : \mathcal{P}(\mathcal{B}) \times \mathcal{P}(\mathcal{B}) \rightarrow \mathcal{P}(\mathcal{B}).$$

Namely $\varphi(P)(I, E)$ is the set of immediate consequences derived by program P starting from a set I of atoms assumed to be true and reacting to the environment perception E.

The mapping φ can be naturally extended to probabilistic interpretations:

$$\varphi_\pi(P) : \ (\mathcal{P}(\mathcal{B}) \times (0,1]) \ \times \ (\mathcal{P}(\mathcal{B}) \times (0,1]) \ \rightarrow \ (\mathcal{P}(\mathcal{B}) \times (0,1])$$

as follows.

Definition 4.4 Let P be a program. For each pair of probabilistic interpretations $\langle I, p\rangle$ and $\langle E, q\rangle$:

$$\varphi_\pi(P)(\langle I, p\rangle, \langle E, q\rangle) = \langle \varphi(P)(I, E) \ , \ p \times q\rangle.$$

□

Namely, given two probabilistic interpretations $\langle I, p\rangle$ and $\langle E, q\rangle$, the mapping $\varphi_\pi(P)$ returns the probabilistic interpretation consisting of the interpretation $\varphi(P)(I, E)$ with associated probability $(p \times q)$. For instance, considering program P of Section 4.1:

$$\varphi_\pi(P)(\langle\{\texttt{Holds(100)}\}, 1\rangle, \langle\{\texttt{Black}\}, .5\rangle) = \langle\{\texttt{Holds(99)}\}, .5\rangle$$
$$\varphi_\pi(P)(\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Red}\}, .5\rangle) = \langle\{\texttt{Holds(100)}\}, .25\rangle$$
$$\varphi_\pi(P)(\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Black}\}, .5\rangle) = \langle\{\texttt{Holds(98)}\}, .25\rangle$$

The reason why the probabilities of I and E are multiplied in order to determine the probability associated with the interpretation $\varphi(P)(I, E)$ will be better understood after introducing the collecting mapping $\mathcal{V}_{\varphi_\pi}$. Intuitively speaking, the reason is that given two *complete* sets of probabilistic interpretations $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$, the result of applying $\varphi_\pi(P)$ to each pair in $(\mathcal{I}_\pi \times \mathcal{E}_\pi)$ should return a *complete* set $\mathcal{J}_\pi$ of probabilistic interpretations, namely $\sum_{\langle J,p\rangle \in \mathcal{J}_\pi} p = 1$.

Typically an agent will start its computation from a complete set of probabilistic interpretations that consists of a single interpretation with associated probability 1. For instance, in the case of the roulette playing agent of Section 4.1, the initial situation in which the agent starts playing with a budget of 100$ corresponds to the set of probabilistic interpretations: $\mathcal{I}_\pi = \{\langle\{\texttt{Holds(100)}\}, 1\rangle\}$.

It is worth noting that the initial set of interpretations may indeed contain more than just one probabilistic interpretation. For instance, determining the amount of the initial budget to bet may be the result of some other reactive computation of the gambling agent. A probabilistic analysis of such a computation may indicate for instance that the initial budget of the agent will be either 100$ or 200$ with the same probability. Such a situation can be represented by considering $\mathcal{I}_\pi = \{\langle\{\texttt{Holds(100)}\}, .5\rangle, \langle\{\texttt{Holds(200)}\}, .5\rangle\}$ as the initial set of probabilistic interpretations.

4.3 Probabilistic behaviours

We now introduce an operator $\mathcal{V}_{\varphi_\pi}$ overs *sets* of probabilistic interpretations in order to collect the set of all possible (one-step) reactive deductions of a program P given a set of probabilistic hypotheses and a set of probabilistic environment perceptions. Formally the operator $\mathcal{V}_{\varphi_\pi}$ maps pairs of sets of probabilistic interpretations into sets of probabilistic interpretations.

Consider again, for instance, program P of section 4.1 together with the probabilistic environment representation $\mathcal{E}_\pi = \{\langle\{\texttt{Red}\}, .5\rangle, \langle\{\texttt{Black}\}, .5\rangle\}$. Intuitively speaking, the set $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ will contain the probabilistic interpretations obtained by applying $\varphi_\pi(P)$ to the probabilistic interpretation $\langle\{\texttt{Holds(100)}\}, 1\rangle$ and to the two environment perceptions in $\mathcal{E}_\pi$. Namely:

$$\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) = \{\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Holds(101)}\}, .5\rangle\}.$$

It is important to note that the mapping $\varphi_\pi(P)$ is not injective, that is, it may well happen that:

$$\varphi_\pi(P)(\langle I_1, p_1\rangle, \langle E_1, q_1\rangle) = \varphi_\pi(P)(\langle I_2, p_2\rangle, \langle E_2, q_2\rangle)$$

for two different pairs of probabilistic interpretations. For this reason, a direct set-theoretic definition of $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ as:

$$\{\langle J, r\rangle \mid \exists \langle E, q\rangle \in \mathcal{E}_\pi, \langle I, p\rangle \in \mathcal{I}_\pi : \langle J, r\rangle = \varphi_\pi(P)(\langle I, p\rangle, \langle E, q\rangle)\}$$

would not properly collect the results of applying $\varphi_\pi(P)$ to all the elements in $(\mathcal{I}_\pi \times \mathcal{E}_\pi)$. For instance, the above definition would imply that:

$$\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\{\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Holds(101)}\}, .5\rangle\}) = \{\langle\{\texttt{Holds(98)}\}, .25\rangle, \langle\{\texttt{Holds(100)}\}, .25\rangle\langle\{\texttt{Holds(102)}\}, .25\rangle\}$$

by incorrectly ignoring that the same probabilistic interpretation

$$\langle\{\texttt{Holds(100)}\}, .25\rangle$$

is produced twice by two different applications of φ_π, viz., by

$$\varphi_\pi(P)(\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Red}\}, .5\rangle)$$

and by

$$\varphi_\pi(P)(\langle\{\texttt{Holds(101)}\}, .5\rangle, \langle\{\texttt{Black}\}, .5\rangle)).$$

We therefore define the operator $\mathcal{V}_{\varphi_\pi}$ in two steps.

(i) We first define a multi-set $\mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ containing all the probabilistic interpretations obtained by applying the φ_π operator to all pairs of probabilistic interpretations in $(\mathcal{I}_\pi \times \mathcal{E}_\pi)$.

(ii) We then define the $\mathcal{V}_{\varphi_\pi}$ operator in terms of the multi-set $\mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$. Intuitively speaking, the set $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ is obtained by transforming the multi-set $\mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ into a set where all the probabilities associated with the same interpretation are summed.

Definition 4.5 Given a program P and two sets $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$ of probabilistic interpretations:

$$\mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) = \{|\ \langle J, r\rangle \mid \exists \langle E, q\rangle \in \mathcal{E}_\pi, \langle I, p\rangle \in \mathcal{I}_\pi : \langle J, r\rangle = \varphi_\pi(P)(\langle I, p\rangle, \langle E, q\rangle)\ |\}$$

$$\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) = \{\langle N, s_N\rangle \mid s_N = \sum\{r \mid \langle J, r\rangle \in \mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)\ \wedge\ J = N\} \wedge s_N > 0\ \}.$$

□

Consider for instance again the gambling agent of Section 4.1. We now have that:

$$\mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\{\langle\{\texttt{Holds(99)}\}, .5\rangle, \langle\{\texttt{Holds(101)}\}, .5\rangle\}) = \{|\ \langle\{\texttt{Holds(98)}\}, .25\rangle, \langle\{\texttt{Holds(100)}\}, .25\rangle, \langle\{\texttt{Holds(100)}\}, .25\rangle, \langle\{\texttt{Holds(102)}\}, .25\rangle\ |\}$$

and:

$$\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\{\langle\{\texttt{Holds(99)}\},.5\rangle,\langle\{\texttt{Holds(101)}\},.5\rangle\}) \;=$$
$$\{\;\langle\{\texttt{Holds(98)}\},.25\rangle,\langle\{\texttt{Holds(100)}\},.5\rangle,\langle\{\texttt{Holds(102)}\},.25\rangle\;\}.$$

The powers of the $\mathcal{V}_{\varphi_\pi}$ operator are defined as usual:

$$\mathcal{V}^0_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi) = \mathcal{I}_\pi$$
$$\mathcal{V}^{n+1}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I})_\pi = \mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)\;(\mathcal{V}^n_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi))$$

4.4 Examples

Example 4.6 Consider again the gambling agent P described in Section 4.1:

```
Holds(S(S(x))) <- Holds(S(x)), Red
Holds(x)       <- Holds(S(x)), Black
Holds(Zero)    <- Holds(Zero)
```

together with the probabilistic environment description:

$$\mathcal{E}_\pi = \{\langle\{\texttt{Red}\},.5\rangle,\langle\{\texttt{Black}\},.5\rangle\}$$

and the initial set of probabilistic interpretations:

$$\mathcal{I}_\pi = \{\langle\{\texttt{Holds(100)}\},1\rangle\}.$$

Given the initial set of probabilistic interpretations $\mathcal{I}_\pi$ and the probabilistic environment description $\mathcal{E}_\pi$, the operator $\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$ will determine the probabilistic behaviours of the gambling agent under the specified hypotheses, as illustrated by Table 1 showing the powers of $\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$.

Table 1

n	$\mathcal{V}^n_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$
0	{ ⟨ { Holds(100) } , 1 ⟩ }
1	{ ⟨ { Holds(99) } , .5 ⟩ , ⟨ { Holds(101) } , .5 ⟩ }
2	{ ⟨ { Holds(98) } , .25 ⟩ , ⟨ { Holds(100) } , .5 ⟩ , ⟨ { Holds(102) } , .25 ⟩ }
3	{ ⟨ { Holds(97) } , .125 ⟩ , ⟨ { Holds(99) } , .375 ⟩ , ⟨ { Holds(101) } , .375 ⟩ , ⟨ { Holds(103) } , .125 ⟩ }
⋮	

□

Example 4.7 Consider again the walking robot described in Example 3.3. The robot used the following program P:

```
(1)     Slow  <- Still, Sun
(2)     Slow  <- Still, Rain
```

```
(3)     Still <- Still, Snow

(4)     Fast  <- Slow, Sun
(5)     Slow  <- Slow, Rain
(6)     Still <- Slow, Snow

(7)     Fast  <- Fast, Sun
(8)     Slow  <- Fast, Rain
(9)     Still <- Fast, Snow
```

to control its walking speed on the base of its perceptions of the weather, where the set of environment perception is $\mathcal{E} = \{$ {Sun},{Rain},{Snow} $\}$.

In order to further analyse the behaviour of the agent, we may make some further assumptions on the probabilities of the environment perceptions. For instance, we may assume that the robot will operate in an environment where snow is a quite rare event, while sun and rain alternate one another with sun slightly prevailing over rain. The assumption that in each moment the weather will be sunny with .5 probability, rainy with .4 probability and snowing with .1 probability is formalised by the probabilistic environment description:

$$\mathcal{E}_\pi = \{\langle \{\texttt{Sun}\}, .5\rangle, \langle \{\texttt{Rain}\}, .4\rangle, \langle \{\texttt{Snow}\}, .1\rangle\}$$

while the assumption that the robot is initially still is modelled by the initial set of probabilistic interpretations:

$$\mathcal{I}_\pi = \{\langle \{\texttt{Still}\}, 1\rangle\}.$$

Given the initial set of probabilistic interpretations $\mathcal{I}_\pi$ and the probabilistic environment description $\mathcal{E}_\pi$, the operator $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ will determine the probabilistic behaviours of our walking robot under the specified hypotheses, as illustrated by Table 2 showing the powers of $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$. Table 2 shows that for the given probabilistic environment description, at any step $n \geq 2$ the robot will be walking slow with probability .45, walking fast with probability .45, and still with probability .1.

Table 2

n	$\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$
0	{ ⟨ { Still } , 1 ⟩ }
1	{ ⟨ { Still } , .1 ⟩ , ⟨ { Slow } , .9 ⟩ }
2	{ ⟨ { Still } , .1 ⟩ , ⟨ { Slow } , .45 ⟩ , ⟨ { Fast } , .45 ⟩ }
3	{ ⟨ { Still } , .1 ⟩ , ⟨ { Slow } , .45 ⟩ , ⟨ { Fast } , .45 ⟩ }
⋮	

The formal characterisation provided by the $\mathcal{V}_{\varphi_\pi}$ operator is particularly useful to reason on different probabilistic behaviours of reactive agents. Suppose for instance that the robot designers want to evaluate whether improving the stability of the robot may effectively lead to having the robot walk faster, given the environment in which it will operate. Simply stated, the question of the engineers is something like: *Is it worth the effort of improving the stability the robot ? Will it really walk faster in that environment ?*

At the software level, re-programming the speed control policy may simply amount to modifying clauses (5), (8) and (9) of the previous program as follows:

```
(1)     Slow  <- Still, Sun
(2)     Slow  <- Still, Rain
(3)     Still <- Still, Snow

(4)     Fast  <- Slow, Sun
(5')    Fast  <- Slow, Rain
(6)     Still <- Slow, Snow

(7)     Fast  <- Fast, Sun
(8')    Fast  <- Fast, Rain
(9')    Slow  <- Fast, Snow
```

The new *possibilistic* behaviours of the agent are synthesised in the graph of Figure 3. Such graph does not however provide an answer to the question of the robot engineers.

Given the initial set of probabilistic interpretations $\mathcal{I}_\pi$ and the probabilistic environment description $\mathcal{E}_\pi$, the operator $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ can be used to determine the *probabilistic* behaviours of the walking robot under the specified hypotheses, as illustrated by Table 3 showing the powers of $\mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$. Table 3 shows that because of the new speed control, the probabilities with which the robot will be walking slow, fast, or be still, will vary at each step. More precisely, the table shows that the larger the number of steps the higher is the probability that the robot will be walking fast.

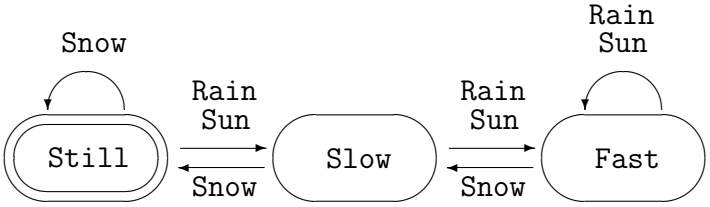

Fig. 3. New possible behaviours of the walking robot

□

Table 3

n	$\mathcal{V}^n_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$
0	{ ⟨ { Still } , 1 ⟩ }
1	{ ⟨ { Still } , .1 ⟩ , ⟨ { Slow } , .9 ⟩ }
2	{ ⟨ { Still } , .1 ⟩ , ⟨ { Slow } , .09 ⟩ , ⟨ { Fast } , .81 ⟩ }
3	{ ⟨ { Still } , .019 ⟩ , ⟨ { Slow } , .171 ⟩ , ⟨ { Fast } , .810 ⟩ }
4	{ ⟨ { Still } , .019 ⟩ , ⟨ { Slow } , .0981 ⟩ , ⟨ { Fast } , .8829 ⟩ }
5	{ ⟨ { Still } , .01171 ⟩ , ⟨ { Slow } , .10539 ⟩ , ⟨ { Fast } , .8829 ⟩ }
6	{ ⟨ { Still } , .01171 ⟩ , ⟨ { Slow } , .098829 ⟩ , ⟨ { Fast } , .889461 ⟩ }
7	{ ⟨ { Still } , .0110539 ⟩ , ⟨ { Slow } , .0994851 ⟩ , ⟨ { Fast } , .889461 ⟩ }
8	{ ⟨ { Still } , .0110539 ⟩ , ⟨ { Slow } , .09889461 ⟩ , ⟨ { Fast } , .89005149 ⟩ }
⋮	

5 Probabilistic analysis of reactive agents

Before discussing the probabilistic analysis of the behaviour of reactive agents, let us present some properties of the $\mathcal{V}_{\varphi_\pi}$ operator introduced in Section 4.3.

5.1 Properties

We first prove that if $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$ are two *complete* sets of probabilistic interpretations, then $\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$ is also a *complete* set of probabilistic interpretations, namely, the probabilities associated with the interpretations in $\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)$ form a probability distribution.

Proposition 5.1 *For each program P, for each complete sets of probabilistic interpretations $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$:*

$$\mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi) \text{ is a complete set of probabilistic interpretations.}$$

Proof. We have to show that:

$$\sum\{p \mid \langle I,p\rangle \in \mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi)\} \;=\; 1.$$

By definition of $\mathcal{M}_{\varphi_\pi}$ we have that:

$$\mathcal{M}_{\varphi_\pi}(P,\mathcal{E}_\pi)(\mathcal{I}_\pi) = \{|\ \langle J,r\rangle \mid \exists \langle E,q\rangle \in \mathcal{E}_\pi, \langle I,p\rangle \in \mathcal{I}_\pi : \langle J,r\rangle = \varphi_\pi(P)(\langle I,p\rangle,\langle E,q\rangle)\ |\}$$

Since $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$ are both complete sets of probabilistic interpretations by hypothesis, we have that:

$$\sum\{p \mid \langle I,p\rangle \in \mathcal{I}_\pi\} = 1 \quad \text{and} \quad \sum\{q \mid \langle E,q\rangle \in \mathcal{E}_\pi\} = 1$$

therefore:

$$\sum\{|\ r \mid \exists\langle E, q\rangle \in \mathcal{E}_\pi, \langle I, p\rangle \in \mathcal{I}_\pi : r = p \times q\ |\} = 1$$

and hence:

$$\sum\{p \mid \langle I, p\rangle \in \mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)\} \ = \ 1.$$

Since for each P, $\mathcal{E}_\pi$ and $\mathcal{J}_\pi$:

$$\sum\{p \mid \langle I, p\rangle \in \mathcal{M}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{J}_\pi)\} \ = \ \sum\{p \mid \langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{J}_\pi)\}$$

we have that:

$$\sum\{p \mid \langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)\} \ = \ 1.$$

□

The immediate corollary of the above proposition is that $\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ is a *complete* set of probabilistic interpretations for each value of n, whenever the environment representation $\mathcal{E}_\pi$ and the initial $\mathcal{I}_\pi$ are *complete* sets of probabilistic interpretations.

Corollary 5.2 *For each program P, for each complete sets of probabilistic interpretations $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$, and for each n:*

$$\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \textit{ is a complete set of probabilistic interpretations.}$$

Proof. The proof is by induction on n.

- (Base case) For $n = 0$ the assertion holds trivially by definition of the powers of $\mathcal{V}_{\varphi_\pi}$, since $\mathcal{V}^0_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) = \mathcal{I}_\pi$ and since $\mathcal{I}_\pi$ is a complete set of probabilistic interpretations by hypothesis.
- (Inductive case) Suppose that $\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ is a complete set of probabilistic interpretations. Then $\mathcal{V}^{n+1}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$ is also a complete set of probabilistic interpretations by Proposition 5.1 since:

$$\mathcal{V}^{n+1}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \ = \ \mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)\ (\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)).$$

□

It is now worth formally establishing the existing tight correspondence between the interpretations computed by the non-probabilistic operator $\mathcal{U}_\varphi$ introduced in Section 3.2 and the probabilistic interpretations computed by the probabilistic operator $\mathcal{V}_\varphi$ introduced in Section 4.3.

Informally, the following Proposition 5.3 states that if an Herbrand interpretation I is obtained by repeatedly applying $\mathcal{V}_{\varphi_\pi}$ to an initial set of probabilistic interpretations $\mathcal{I}_\pi$, then the same interpretation I can be obtained by applying the same number of times $\mathcal{U}_\varphi$ to the set of Herbrand interpretations contained in $\mathcal{I}_\pi$. To simplify notation, we will denote by $\mathcal{I}_{\not\pi}$ the set of Herbrand interpretations contained in a set $\mathcal{I}_\pi$ of probabilistic interpretations, that is: $\mathcal{I}_{\not\pi} = \{I \mid \langle I, p\rangle \in \mathcal{I}_\pi\}$.

Conversely, Proposition 5.4 establishes that any Herbrand interpretation obtained by repeatedly applying $\mathcal{U}_\varphi$ to an initial set of interpretations $\mathcal{I}$ is also obtained by applying the same number of times $\mathcal{V}_{\varphi_\pi}$ to any initial set of probabilistic interpretations $\mathcal{I}_\pi$ such that $\mathcal{I}_{\not\pi} = \mathcal{I}$.

Proposition 5.3 *For each program P, for each sets of probabilistic interpretations $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$, and for each n:*

$$\langle I, p\rangle \in \mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \implies I \in \mathcal{U}^n_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{I}_{\not\pi})$$

Proof. The proof is by induction on n:

- (Base case) The base case ($n = 0$) trivially holds by definition of the powers of $\mathcal{V}_{\varphi_\pi}$ and $\mathcal{U}_\varphi$:

  $\langle I, p\rangle \in \mathcal{V}^0_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$
  $\Leftrightarrow$ { by definition of powers of $\mathcal{V}_{\varphi_\pi}$ }
  $\langle I, p\rangle \in \mathcal{I}_\pi$
  $\Rightarrow$ { by definition of $\mathcal{I}_{\not\pi}$ }
  $I \in \mathcal{I}_{\not\pi}$
  $\Leftrightarrow$ { by definition of powers of $\mathcal{U}_\varphi$ }
  $I \in \mathcal{U}^0_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{I}_{\not\pi})$.

- (Inductive case)

  $\langle I, p\rangle \in \mathcal{V}^{n+1}_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$
  $\Leftrightarrow$ { by definition of powers of $\mathcal{V}_{\varphi_\pi}$ }
  $\langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}(P, \mathcal{E}_\pi)\ (\mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi))$
  $\Rightarrow$ { by definition of $\mathcal{V}_{\varphi_\pi}$ }
  $\exists \langle E, q\rangle \in \mathcal{E}_\pi, \langle J, r\rangle \in \mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi), s : \quad \langle I, s\rangle = \varphi_\pi(P)(\langle J, r\rangle, \langle E, q\rangle)$
  $\Rightarrow$ { by definition of φ_π }
  $\exists \langle E, q\rangle \in \mathcal{E}_\pi, \langle J, r\rangle \in \mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) : \quad I = \varphi(P)(J, E)$
  $\Rightarrow$ { by definition of $\mathcal{E}_{\not\pi}$ and by inductive hypothesis }
  $\exists E \in \mathcal{E}_{\not\pi}, J \in \mathcal{U}^n_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{I}_{\not\pi}) : \quad I = \varphi(P)(J, E)$
  $\Rightarrow$ { by definition of $\mathcal{U}_\varphi$ }
  $I \in \mathcal{U}_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{U}^n_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{I}_{\not\pi}))$
  $\Leftrightarrow$ { by definition of powers of $\mathcal{U}_\varphi$ }
  $I \in \mathcal{U}^{n+1}_\varphi(P, \mathcal{E}_{\not\pi})(\mathcal{I}_{\not\pi})$

□

Proposition 5.4 *For each program P, let $\mathcal{E}$ and $\mathcal{I}$ be two sets of interpretations and let $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$ be two sets of probabilistic interpretations such that $\mathcal{E}_{\not\pi} = \mathcal{E}$ and $\mathcal{I}_{\not\pi} = \mathcal{I}$. Then for each n:*

$$I \in \mathcal{U}^n_\varphi(P, \mathcal{E})(\mathcal{I}) \implies \exists p : \ \langle I, p\rangle \in \mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi)$$

Proof. The proof is by induction on n:

- (Base case) The base case ($n = 0$) trivially holds by definition of the powers of $\mathcal{V}_{\varphi_\pi}$ and $\mathcal{U}_\varphi$:

$$
\begin{array}{ll}
 & I \in \mathcal{U}_{\varphi}^{0}(P,\mathcal{E})(\mathcal{I}) \\
\Leftrightarrow & \quad \{ \text{ by definition of powers of } \mathcal{U}_{\varphi} \ \} \\
 & I \in \mathcal{I} \\
\Rightarrow & \quad \{ \text{ since } \mathcal{I}_{\not\sharp} = \mathcal{I} \ \} \\
 & \exists p : \ \langle I, p\rangle \in \mathcal{I}_{\pi} \\
\Leftrightarrow & \quad \{ \text{ by definition of powers of } \mathcal{V}_{\varphi_\pi} \ \} \\
 & \exists p : \ \langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}^{0}(P,\mathcal{E}_{\pi})(\mathcal{I}_{\pi})
\end{array}
$$

- (Inductive case)

$$
\begin{array}{lll}
 & I \in \mathcal{U}_{\varphi}^{n+1}(P,\mathcal{E})(\mathcal{I}) & \\
\Leftrightarrow & \quad \{ \text{ by definition of powers of } \mathcal{U}_{\varphi} \ \} & \\
 & I \in \mathcal{U}_{\varphi}(P,\mathcal{E}) \ (\mathcal{U}_{\varphi}^{n}(P,\mathcal{E})(\mathcal{I})) & \\
\Rightarrow & \quad \{ \text{ by definition of } \mathcal{U}_{\varphi} \ \} & \\
 & \exists E \in \mathcal{E}, J \in \mathcal{U}_{\varphi}^{n}(P,\mathcal{E})(\mathcal{I}) : & I = \varphi(P)(J,E) \\
\Rightarrow & \quad \{ \text{ since } \mathcal{E}_{\not\sharp} = \mathcal{E} \text{ and by inductive hypothesis } \} & \\
 & \exists \langle E, q\rangle \in \mathcal{E}_{\pi}, \langle J, r\rangle \in \mathcal{V}_{\varphi_\pi}^{n}(P,\mathcal{E}_{\pi})(\mathcal{I}_{\pi}) : & I = \varphi(P)(J,E) \\
\Rightarrow & \quad \{ \text{ by definition of } \varphi_{\pi} \ \} & \\
 & \exists \langle E, q\rangle \in \mathcal{E}_{\pi}, \langle J, r\rangle \in \mathcal{V}_{\varphi_\pi}^{n}(P,\mathcal{E}_{\pi})(\mathcal{I}_{\pi}), p : & \langle I, p\rangle = \varphi_{\pi}(P)(\langle J, r\rangle, \langle E, q\rangle) \\
\Rightarrow & \quad \{ \text{ by definition of } \mathcal{V}_{\varphi_\pi} \ \} & \\
 & \exists p : \ \langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}(P,\mathcal{E}_{\pi}) \ (\mathcal{V}_{\varphi_\pi}^{n}(P,\mathcal{E}_{\pi})(\mathcal{I}_{\pi})) & \\
\Leftrightarrow & \quad \{ \text{ by definition of powers of } \mathcal{V}_{\varphi_\pi} \ \} & \\
 & \exists p : \ \langle I, p\rangle \in \mathcal{V}_{\varphi_\pi}^{n+1}(P,\mathcal{E}_{\pi})(\mathcal{I}_{\pi}) &
\end{array}
$$

□

5.2 Probabilistic beliefs

The availability of a formal characterisation of the probabilistic behaviours of reactive programs provides a firm ground for reasoning about them.

One of the properties of interest in the practice of multi-agent systems is which are the conclusions that the agent may draw by reacting to the external environment, the evolution of the latter being *a priori* unknown. The notion of *possible beliefs* was introduced in [4] to formally characterise the set of conclusions that a program P may possibly draw, starting with an initial set of interpretations $\mathcal{I}$ and reacting to a set of environment perceptions $\mathcal{E}$. An effective, resource bounded characterisation of the possible beliefs of a program after n steps of computation can be formalised as follows.

Definition 5.5 Let P be a program, let $\mathcal{I}$ and $\mathcal{E}$ be two sets of interpretations, and let $n > 0$. We put:

$$\mathrm{PB}(P,\mathcal{E},\mathcal{I},n) \ = \ \{A \mid I \in \mathcal{U}_{\varphi}^{n}(P,\mathcal{E})(\mathcal{I}) \ \wedge \ A \in I\}.$$

□

Namely $\mathrm{PB}(P,\mathcal{E},\mathcal{I},n)$ denotes the set of conclusions that program P *may* possibly derive after n steps, starting with an initial set of interpretations $\mathcal{I}$

and reacting to a set of environment perceptions $\mathcal{E}$.

For instance, for the walking robot described in Example 3.3, we have that $\mathrm{PB}(P, \mathcal{E}, \mathcal{I}, n) = \{\texttt{Still}, \texttt{Slow}, \texttt{Fast}\}$ for each $n \geq 2$. Notice that possible beliefs are defined in terms of the possibilistic collecting operator $\mathcal{U}_{\varphi}$, which does not take into account the probabilities that may be associated with environment perceptions. For instance, while in Example 4.7 we have shown that a revised control speed program for the robot does exhibit different probabilistic behaviours, the two programs have the same set of possible beliefs.

We therefore introduce the notion of *probabilistic beliefs* of a program P that starts with an initial set of probabilistic interpretations $\mathcal{I}_{\pi}$ and reacts to a set of probabilistic perceptions $\mathcal{E}_{\pi}$. Probabilistic beliefs associate each possible belief with a corresponding probability as formalised by the following definition.

Definition 5.6 Let P be a program, let $\mathcal{I}_{\pi}$ and $\mathcal{E}_{\pi}$ be two sets of probabilistic interpretations, and let $n > 0$. We put:

$$\Pi\mathrm{B}(P, \mathcal{E}_{\pi}, \mathcal{I}_{\pi}, n) \;=\; \{(A, p) \mid p = \textstyle\sum\{q \mid \langle I, q\rangle \in \mathcal{V}^{n}_{\varphi_{\pi}}(P, \mathcal{E}_{\pi})(\mathcal{I}_{\pi}) \wedge A \in I\} \;\wedge\; p > 0\}.$$

□

Namely $\Pi\mathrm{B}(P, \mathcal{E}_{\pi}, \mathcal{I}_{\pi}, n)$ denotes the set of atoms that program P may derive in n steps starting from $\mathcal{I}_{\pi}$ and reacting to $\mathcal{E}_{\pi}$. The probability associated with each atom A is the sum of the probabilities of all the probabilistic interpretations in which A occur.

For instance, in the case of the walking robot described in Example 3.3, the set of probabilistic beliefs after 10 steps is:

$$\Pi\mathrm{B}(P, \mathcal{E}_{\pi}, \mathcal{I}_{\pi}, 10) = \{(\texttt{Still}, .1), (\texttt{Slow}, .45), (\texttt{Fast}, .45)\}.$$

Notice that, as expected, the notion of probabilistic beliefs is able to distinguish possibilistic behaviours that are probabilistically different. For instance, the revised control speed program described in 4.7 does have a different set of probabilistic beliefs:

$$\Pi\mathrm{B}(Q, \mathcal{E}_{\pi}, \mathcal{I}_{\pi}, 10) = \{\; (\texttt{Still}, .010994851), (\texttt{Slow}, .0989005149), (\texttt{Fast}, .8901046341)\}.$$

The notion of probabilistic beliefs extends the notion of possible beliefs by associating a probability to each possible belief. The relation between the two notions is formally stated by the following corollary.

Corollary 5.7 *For each program P, let $\mathcal{E}$ and $\mathcal{I}$ be two sets of interpretations and let $\mathcal{E}_{\pi}$ and $\mathcal{I}_{\pi}$ be two sets of probabilistic interpretations such that $\mathcal{E}_{\not\pi} = \mathcal{E}$ and $\mathcal{I}_{\not\pi} = \mathcal{I}$. Then for each n:*

$$\{A \mid (A, p) \in \Pi B(P, \mathcal{E}_{\pi}, \mathcal{I}_{\pi}, n)\} \;=\; PB(P, \mathcal{E}, \mathcal{I}, n).$$

Proof. The corollary descends from Propositions 5.3 and 5.4.

$$
\begin{array}{ll}
& B \in \{A \mid (A,p) \in \Pi\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n)\} \\
\Leftrightarrow & \\
& \exists p: \ (B,p) \in \Pi\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \\
\Leftrightarrow & \quad \{ \text{ by definition of probabilistic beliefs } \} \\
& \exists q, I: \ \langle I, q \rangle \in \mathcal{V}^n_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \wedge B \in I \\
\Leftrightarrow & \quad \{ \text{ by Propositions 5.3 and 5.4 } \} \\
& \exists I: \ I \in \mathcal{U}^n_{\varphi}(P, \mathcal{E})(\mathcal{I}) \wedge B \in I \\
\Leftrightarrow & \quad \{ \text{ by definition of possible beliefs } \} \\
& B \in \mathrm{PB}(P, \mathcal{E}, \mathcal{I}, n)
\end{array}
$$

□

Other useful notions can be defined in terms of the probabilistic beliefs. For instance, when analysing the probabilistic behaviours of a reactive agent, one is often more interested in determining the "most probable" beliefs of the agents rather than in examining the full set of probabilistic beliefs.

More precisely, we are typically interested in determining the probabilistic beliefs whose associated probability is above a given belief threshold t. The following corresponding notion of *likely beliefs* is defined as a natural refinement of the notion of probabilistic beliefs.

Definition 5.8 Let P be a program, let $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$ be two sets of probabilistic interpretations, and let $n > 0$. We put:

$$\mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{t}) \ = \ \{A \mid (A,p) \in \Pi\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \wedge p \geq \mathbf{t}\}$$

□

It is easy to observe that the likely beliefs are a subset of the probabilistic beliefs of an agent since for a generic threshold $\mathbf{t}$:

$$\mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{t}) \ \subseteq \ \{A \mid (A,p) \in \Pi\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n)\} \ = \ \mathrm{PB}(P, \mathcal{E}, \mathcal{I}, n)$$

while for $\mathbf{t} = 0$:

$$\mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{0}) \ = \ \{A \mid (A,p) \in \Pi\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n)\} \ = \ \mathrm{PB}(P, \mathcal{E}, \mathcal{I}, n).$$

It is worth observing that the notion of likely beliefs gives a handy representation of a subset of the probabilistic beliefs of practical use. Consider for instance again the problem of comparing the probabilistic behaviours of the walking robot when varying its speed control program. If the robot engineers consider .8 as their belief threshold, then they obtain:

$$\mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, 10, .\mathbf{8}) = \{\}$$

for the first control speed program of Example 3.3, and:

$$\mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, 10, .\mathbf{8}) = \{\texttt{Fast}\}$$

for the revised control speed program described of Example 4.7.

A further notion of practical utility is a specialisation of the notion of likely beliefs. Simply stated, when performing a probabilistic analysis of the agent behaviours, we are interested in determining whether the programs will have some definite beliefs.

Definition 5.9 Let P be a program, let $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$ be two sets of probabilistic interpretations, and let $n > 0$. We put:

$$\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \;=\; \mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{1}).$$

□

We will see in the following section that the beliefs of a program are the basis to define the important notion of probabilistic invariant.

5.3 Probabilistic invariant

The invariant of a conventional program defines the properties that hold at each stage of the program computation. Analogously, the invariant of a reactive program defines the largest set of conclusions that the program will be able to draw at any time in any environment. An effective, resource bounded characterisation of the invariant of a program after n steps of computation can be formalised as follows.

Definition 5.10 Let P be a program, let $\mathcal{I}$ and $\mathcal{E}$ be two sets of interpretations, and let $n > 0$. We put:

$$\mathrm{Inv}(P, \mathcal{E}, \mathcal{I}, n) \;=\; \{A \mid \forall i \in [1, n] :\; (I \in \mathcal{U}^i_\varphi(P, \mathcal{E})(\mathcal{I}) \implies A \in I)\}.$$

□

Namely an atom A belongs to the invariant $\mathrm{Inv}(P, \mathcal{E}, \mathcal{I}, n)$ if A belongs to all the interpretations computed by $\mathcal{U}_\varphi$ during all the first n computation steps. The notion of invariant can be reformulated in a probabilistic setting as follows.

Definition 5.11 Let P be a program, let $\mathcal{I}_\pi$ and $\mathcal{E}_\pi$ be two sets of probabilistic interpretations, and let $n > 0$. We put:

$$\mathrm{Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \;=\; \{A \mid \forall i \in [1, n] :\; (\langle I, p\rangle \in \mathcal{V}^i_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \implies A \in I)\}.$$

□

It is worth observing that the notion of invariant can be equivalently formulated in terms of the beliefs of a program (see Definition 5.9). Namely, for each sets $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$ of probabilistic interpretations, and for each n:

$$\mathrm{Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \;=\; \bigcap_{i \in [1,n]} B(P, \mathcal{E}_\pi, \mathcal{I}_\pi, i) \;=\; \bigcap_{i \in [1,n]} \mathcal{L}\mathrm{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{1}).$$

It is important to observe that two definitions of invariant (Definitions 5.10 and 5.11) coincide, as stated by the following proposition.

Proposition 5.12 *For each program P, let $\mathcal{E}$ and $\mathcal{I}$ be two sets of interpretations and let $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$ be two sets of probabilistic interpretations such that $\mathcal{E}_{\not\pi} = \mathcal{E}$ and $\mathcal{I}_{\not\pi} = \mathcal{I}$. Then for each n:*

$$Inv(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \;=\; Inv(P, \mathcal{E}, \mathcal{I}, n).$$

Proof.

$A \in \mathrm{Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n)$
$\Leftrightarrow$ { by Definition 5.11 }
$\forall i \in [1,n] : \; (\langle I, p\rangle \in \mathcal{V}^i_{\varphi_\pi}(P, \mathcal{E}_\pi)(\mathcal{I}_\pi) \implies A \in I)\}$
$\Leftrightarrow$ { by Propositions 5.3 and 5.4 }
$\forall i \in [1,n] : \; (I \in \mathcal{U}^i_{\varphi}(P, \mathcal{E})(\mathcal{I}) \implies A \in I)\}$
$\Leftrightarrow$ { by Definition 5.10 }
$A \in \mathrm{Inv}(P, \mathcal{E}, \mathcal{I}, n)$.

□

The above proposition implicitly suggests that notion of invariant of Definition 5.11 is not really probabilistic.

Example 5.13 Consider again the walking robot discussed in Examples 3.3 and 3.5. Suppose that the robot designers have discovered that their robot does not resist for a long time under the snow. The question of the engineers is now: *Will the robot survive in the environment where it is supposed to operate? Or should we rather improve the robot resistance to ice to be reasonably certain that it will not fall KO?*

The new possible behaviours of the robot are specified by the following program P:

```
Slow     <- Still, Sun          Fast     <- Slow, Sun
Slow     <- Still, Rain         Fast     <- Slow, Rain
Freezing <- Still, Snow         Still    <- Slow, Snow
OK       <- Still, Sun          OK       <- Slow, Sun
OK       <- Still, Rain         OK       <- Slow, Rain
OK       <- Still, Snow         OK       <- Slow, Snow

Fast     <- Fast, Sun           Still    <- Freezing, Sun
Fast     <- Fast, Rain          Freezing <- Freezing, Rain
Slow     <- Fast, Snow          KO       <- Freezing, Snow
OK       <- Fast, Sun           OK       <- Freezing, Sun
OK       <- Fast, Rain          OK       <- Freezing, Rain
OK       <- Fast, Snow          KO       <- KO
```

According to the above specification, answering the engineers question reduces to checking that the program invariant contains the atom `OK`.

Let us consider the same probabilistic environment description employed in Examples 3.3 and 3.5:

$$\mathcal{E}_\pi = \{\langle\{\texttt{Sun}\}, .5\rangle, \langle\{\texttt{Rain}\}, .4\rangle, \langle\{\texttt{Snow}\}, .1\rangle\}.$$

and the initial assumption that the robot is initially still and ok:

$$\mathcal{I}_\pi = \{\langle\{\texttt{Still}, \texttt{OK}\}, 1\rangle\}.$$

It is easy to observe that the program has an empty invariant $\text{Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n)$ for each $n > 1$. Indeed, the robot *may possibly* reach the KO state if there is a sequence of snow perceptions. On the other hand, probabilistically speaking, if the probability that such an event will occur is low enough, we may consider OK a "probabilistic" invariant of the robot. □

Following the above example, we introduce the notion of *probabilistic invariant* which relates the condition of invariance to a belief threshold. Informally speaking, an atom belongs to the probabilistic invariant of a program if it belongs to the likely beliefs of the program for all the first n computation steps.

Definition 5.14 For each program P, let $\mathcal{E}_\pi$ and $\mathcal{I}_\pi$ be two sets of probabilistic interpretations. Then for each n:

$$\Pi\text{-Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{t}) \;=\; \bigcap_{i \in [1,n]} \mathcal{L}\text{B}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{t}).$$

□

It is easy to observe that the notion of probabilistic invariant generalises the notion of invariant of Definition 5.11, as:

$$\text{Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n) \;=\; \Pi\text{-Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, n, \mathbf{1})$$

Example 5.15 Consider again the question of the robot engineers discussed in Example 5.13. We observe that while the (non-probabilistic) invariant of the program is empty for any $n > 1$, the notion of probabilistic invariant supports a real probabilistic analysis of the robot behaviours. For instance, if the robot engineers choose .95 as their belief threshold for the probabilistic invariant, then they obtain:

$$\Pi\text{-Inv}(P, \mathcal{E}_\pi, \mathcal{I}_\pi, 100, .\mathbf{95}) \;=\; \{\texttt{OK}\}.$$

□

6 Related work

The idea of modeling different types of reactive computations as variants of standard logic programming bottom-up computations is, to the best of our

knowledge, original. Nevertheless, our work relates to a number of other approaches proposed in the literature.

Our representation of all possible computations bears strong similarities with the possible worlds semantics of modal logics [9]. The notion of invariant defines the set of (atomic) formulae which are "necessarily" true (i.e., true in each possible world), while the set of possible beliefs defines the set of (atomic) formulae which are "possibly" true (i.e., true in at least one possible world). In spite of these similarities, our approach obviously differs from modal logics, which address other issues such as temporal reasoning or introspection. On the other hand, our semantics based on logic programming is simpler than the semantics of modal logics, and it accounts for an effective implementation as it is a proper generalisation of the standard semantics used in deductive data bases applications [17].

Several efforts have been devoted to investigate the role of computational logic in multi-agent systems (see [14] for a quite recent road map). The IMPACT system [16] is one of the best known examples of multi-agent system relying on computational logic. Agent beliefs are represented by agent programs, in the style of logic programming, while integrity constraints and action bases are used to describe the actions that individual agents can perform. A number of interesting applications of IMPACT have been illustrated, including its recent extension to deal with temporal reasoning [8]. Several other efforts have focussed on the use of computational logic to represent incomplete information in multi-agent systems. The use of abduction to represent incomplete communication environments [15], and the use of updates to represent dynamically evolving knowledge [2,12] are two promising approaches that have been recently proposed. The agent-based architecture presented in [11] aims at reconciling rationality and reactivity, and it is probably the work most related to ours. In [11] agents are logic programs which continuously perform an "observe-think-act" cycle, and their behaviour is defined via a proof procedure which exploits iff-definitions and integrity constraints.

A considerable amount of work has been devoted in the logic programming community to model open programs and their composition (see [5] for a survey). While these works share with ours the adoption of the logic programming paradigm as specification language, they focus on the composition of static programs. In contrast, we focus on the composition of a program with an external dynamic environment, and analyse its reactive, incremental computations.

Abductive logic programming [10] is another approach to modeling incomplete knowledge. In this setting, agents may abduce external hypotheses provided that they satisfy existing integrity constraints. While we focus on bottom-up semantics, abductive logic programming is defined via proof procedures which combine backward reasoning with integrity constraint checking. A promising direction for further developments is to employ abduction to express forms of interaction among agents, as indicated in [6,7].

Finally, a large body of research has been devoted in the last two decades to the study of concurrency. The main focus of these activities is to model process interactions abstracting from internal computations steps. In contrast our focus is on the interplay between interaction and computation. It is worth mentioning that [13] explicitly introduces a notion of external environment which resembles ours, even if in a quite different context. Also the semantics of interaction presented in [1] employs an explicit representation of the environment. That semantics is formulated in categorical terms and it is based on linear logic and game semantics.

7 Concluding remarks

We have presented a simple logic-based formalization of the behaviours of agents capable of reacting to changes occurring in the external environment. In particular we have focussed on modeling the probabilistic behaviours of agents reacting to environment perceptions with an associated probability distribution. We have shown how the availability of a formal characterisation supports effective, resource bounded, quantitative analyses of the probabilistic behaviours of reactive agents, as illustrated by the notions of probabilistic beliefs and probabilistic invariants discussed in Section 5.

Our formalisation does not however account for several aspects of multi-agent systems. We have not considered, for instance, the way in which an agent may affect the behaviour of the environment. Indeed, a natural extension of our setting is to include the "act" part of the "observe-think-act" cycle [11] by employing an intensional representation of the environment. This may be done by representing the environment as a set of transformations $T_i(I)$ over interpretations, rather than as a set of interpretations, and by enabling agents to affect the interpretation I from which the environment evolves. Another important extension is the introduction of negation in our setting. Indeed the ability to deal with incomplete information is another important aspect in multi-agent systems. The use of updates to represent dynamically evolving knowledge described in [2] seems a promising approach in this direction. These extensions are scope for future work.

References

[1] Abramsky, S., *Semantics of Interaction: an Introduction to Game Semantics*, in: P. Dybjer and A. Pitts, editors, *Proceedings of the CLiCS 96 Summer School* (1997), pp. 1–31.

[2] Alferes, J., J. Leite, L. Pereira, H. Przymusinska and T. Przymusinski, *Dynamic updates of non-monotonic knowledge bases*, Journal of Logic Programming **45(1-3)** (2000), pp. 43–70.

[3] Apt, K. R., *Logic programming*, in: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science* (1990), pp. 493–574, vol. B.

[4] Brogi, A., S. Contiero and F. Turini, *On the interplay between reactivity and rationality in multi-agent systems*, Technical report, Department of Computer Science, University of Pisa (2001).

[5] Bugliesi, M., E. Lamma and P. Mello, *Modularity in Logic Programming*, Journal of Logic Programming **19-20** (1992), pp. 443–502.

[6] Ciampolini, A., E. Lamma, P. Mello and P. Torroni, *Expressing collaborative and competitive coordination among abductive logic agents*, in: F. Sadri and K. Satoh, editors, *Proceedings of the CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA'00)*, 2000, pp. 35–43.

[7] Dell'Acqua, P., F. Sadri and F. Toni, *Combining introspection and communication with rationality and reactivity in agents*, in: J. Dix, L. F. del Cerro and U. Furbach, editors, *Logics in Artificial Intelligence, European Workshop, JELIA '98, LNCS 1489* (1998), pp. 17–32.

[8] Dix, J., S. Kraus and V. Subrahmanian, *Temporal Agent Reasoning*, Artificial Intelligence **127** (2001), pp. 87–135.

[9] Hughes, G. and M. Cresswell, "A New Introduction to Modal Logic," RoutLedge, 1996.

[10] Kakas, A., R. Kowalski and F. Toni, *Abductive logic programming*, Journal of Logic and Computation **2(6)** (1992), pp. 719–770.

[11] Kowalski, R. and F. Sadri, *Towards a Unified Agent Architecture that Combines Rationality and Reactivity*, in: C. Zaniolo and D. Pedreschi, editors, *Logic in Databases* (1996), pp. 137–150.

[12] Leite, J. A., J. J. Alferes and L. M. Pereira, *Multi-dimensional dynamic logic programming*, in: F. Sadri and K. Satoh, editors, *Proceedings of the CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA'00)*, 2000, pp. 17–26.

[13] Letichevsky, A. and D. Gilbert, *A general theory of action languages*, Cybernetics and System Analysis **1** (1998), pp. 16–37.

[14] Sadri, F. and F. Toni, *Computational logic and multi-agent systems: a roadmap* (1999).
URL `http://www.compulog.org`

[15] Satoh, K., K. Inoue, K. Iwanuma and C. Sakama, *Speculative computation by abduction under incomplete communication environments*, in: *Proceedings of the Fourth International Conference on Multi-Agent Systems*, 2000, pp. 263–270.

[16] Subrahmanian, V., P. Bonatti, J. Dix, T. Eiter, S. K. F. Ozcan and R. Ross, "Heterogeneous agent systems," MIT Press, 2000.

[17] Zaniolo, C., S. Ceri, C. Faloustos, R. Snodgrass, V. Subrahmanian and R. Zicari, "Advanced Database Systems," Morgan Kaufmann, 1997.