

# Logical Implementation of Uncertain Agents

Nivea de C. Ferreira, Michael Fisher, and Wiebe van der Hoek

Department of Computer Science, University of Liverpool, UK  
{niveacf, michael, wiebe}@csc.liv.ac.uk

**Abstract.** We consider the representation and execution of agents specified using *temporal logics*. Previous work in this area has provided a basis for the direct execution of agent specifications, and has been extended to allow the handling of agent beliefs, deliberation and multi-agent groups. However, the key problem of *uncertainty* has not been tackled. Given that agents work in unknown environments, and interact with other agents that may, in turn, be unpredictable, then it is essential for any formal agent description to incorporate some mechanism for capturing this aspect. Within the framework of executable specifications, formal descriptions involving uncertainty must also be executable. The contribution of this paper is to extend executable temporal logic in order to allow the representation and execution of uncertain statements within agents. In particular, we extend the basis of the METATEM temporal framework with a *probabilistic belief* dimension captured by the recently introduced  $P_FKD45$  logic. We provide a description of the extended logic, the translation procedure for formulae in this extended logic to an executable normal form, and the execution algorithm for such formulae. We also outline technical results concerning the correctness of the translation to the normal form and the completeness of the execution mechanism.

## 1 Introduction

The logical characterisation of agent-based systems is now a well established area [6,19,21]. Such a characterisation can not only provide an unambiguous semantics for agents but can also allow key techniques such as formal specification to be used in the analysis of agent-based systems. An important aspect is the direct execution of these formal agent specifications. Here, a model satisfying the agent specification is extracted, with the process of extracting such a model corresponding to an execution. This is analogous to the use of Prolog in classical logic whereby the system searches for a model satisfying the specification. Such direct execution also provides a strong link between the semantics of an agent and its implementation, something that is often lacking in contemporary agent programming frameworks [5].

Given that agents essentially work in unpredictable environments, and interact with other agents that may, in turn, be unpredictable, it is vital for any formal description of an agent to incorporate some mechanism for capturing uncertainty [14] on a level that is more fine-grained than assuming the agent has just knowledge, or even beliefs. In dealing with uncertainty one can choose among many existing approaches, often categorised as either numerical or symbolic. We here opt for a framework incorporating *Probability Theory* as, at least semantically, this allows us to remain close to the *possible worlds paradigm*. This possible-worlds view is by far the most popular way (see also [20]) to model the varieties of agent we require (cf. [19]).

In our work, we choose to extend a standard temporal logic, to which execution has previously been applied, with an added element of uncertainty. Thus, we produce a new logic from the combination of standard temporal logic with a novel logic of *probabilistic belief*,  $P_FKD45$  [8]. This provides a simple, and intuitive, basis for specifying agents uncertain about their environment and their choices.

Once we have the temporal specification of an uncertain agent, an implementation can be developed in a number of ways, for example by refinement to a standard programming language or the automatic synthesis of an automaton [17,16]. Such a synthesis approach is necessarily complex, generating an implementation that is guaranteed to satisfy the specification in all environments. However, the route we choose is to directly execute the temporal specification in order to provide an implementation. Note that this just involves searching for *one* acceptable execution, which is generally less complex (and often much quicker) than attempting full synthesis [2]. Our approach to direct execution extends the METATEM programming language [3], which executes purely temporal statements, and can be utilised to animate agent specifications.

The contribution of this paper is to devise a logical framework for uncertainty in agents that (1) is conceptually clear and simple, (2) can be used in modal (intensional) logical specifications of agents, and (3) can be added on top of METATEM to give a rich but still executable temporal/doxastic logic. The executable framework we develop is called PROTEM.

The rest of this paper is organised in the following way. Section 2 presents a brief overview of the METATEM Framework and the  $P_FKD45$  logic. We then describe how these systems can be combined forming the basis for uncertain agent implementation, with the associated normal form and execution mechanism being subsequently described in Sections 3 and 4, respectively. Finally, in Section 5, comments on related work, potential applications, and future research, together with concluding remarks, are provided.

## 2 A Temporal Doxastic Logic

### 2.1 Temporal Basis of METATEM

In previous work on the METATEM framework, the representation of simple dynamic agents using temporal logic [12], the representation of deliberation within these agents, and an extension to agents that have beliefs [10] were considered. In the work presented in this paper we provide an extension of this framework, using an appropriate logical formalism to incorporate uncertainty.

In METATEM, logical formulae represent an agent's specification. The framework allows the animation of an agent's specification by direct execution of these formulae, essentially providing an implementation of the agent's behaviour. This approach follows the *imperative future* paradigm [4], and applies an iterative *forward-chaining* process to a set of temporal formulae in a specific normal form in order to (attempt to) construct a model for the specification.

Temporal logic is an extension of classical logic in which temporal order is important. Thus, statements are not just true or false, but are true or false dependent upon the moment in time at which they are evaluated. Typical operators of the temporal logic

used are ‘ $\circ$ ’ (“in the next moment in time”), ‘ $\diamond$ ’ (“at some point in the future”), ‘ $\square$ ’ (“always in the future”), ‘ $U$ ’ (“until”) and ‘**start**’ (“at the start”), e.g. ‘ $\diamond happy$ ’.

Since the underlying temporal model is a linear, discrete sequence of states, and since forward chaining is applied, model construction in this way mimics execution in more standard programming languages [12]. This approach is captured in the METATEM programming language [3], where execution involves forward chaining via temporal formulae from initial conditions, while constraining the execution in an attempt to satisfy eventualities. The underlying mechanism here is relatively straightforward, forward chaining through formulae of the form

$$antecedent (present) \Rightarrow consequent (future)$$

attempting to construct a model. If a disjunction, such as  $red \vee blue$  is executed, a choice must be made. If this choice leads to a contradictory situation, then backtracking occurs and an alternative choice is made.

The main complication comes from execution of  $\diamond$ -formulae (or ‘eventualities’). When a formula, such as  $\diamond\varphi$ , is executed, the system must attempt to ensure that  $\varphi$  *eventually* becomes true. As such eventualities might not be able to be satisfied immediately, a record of the outstanding eventualities must be kept, so that they can be re-tried as execution proceeds. The standard heuristic used is to attempt to satisfy as many eventualities as possible, starting with the *oldest outstanding eventuality* [4]. This helps ensure fairness and completeness of the execution mechanism.

*Remark 1.* Although we will not consider it here, it is important to note that, just as in Logic Programming, the backtracking nature of individual agents must be modified when situated in a multi-agent environment. Typically, agents are allowed to backtrack, but *not* past the point where the agent affects its environment. Sending a message, for example to other agents, effectively acts as a “cut” operation on the search space. Thus, agents are often programmed to have phases of “thinking” (using backtracking) interspersed with phases of communication.

## 2.2 Probabilistic Doxastic Logic

While temporal logic is used to capture the dynamic behaviour of an agent, *modal logics* can be used to extend this basic framework with beliefs, abilities, etc.  $P_FKD45$  [8]<sup>1</sup> is a complete, compact and conceptually simple modal formalism for probabilistic reasoning. In particular, it is useful for representing and reasoning about (static) uncertainty within computational agents. The language of  $P_FKD45$  consists of a countable set of propositional symbols, the logical connectives  $\neg$  and  $\vee$  (with standard definitions for  $\perp$ ,  $\top$ ,  $\wedge$ ,  $\Rightarrow$ ,  $\Leftrightarrow$ ), and parentheses. Its basic modal operator is  $P_x^>$  (where  $x$  is a rational number within the interval  $[0, 1]$ ), and the intended meaning of  $P_x^>\varphi$  is:

$$\varphi \text{ is believed to have a probability strictly greater than } x$$

<sup>1</sup> Interested readers will find in this paper a more detailed comparison between  $P_FKD45$  and other Probabilistic Logics.

The operators  $P_x^{\geq}$ ,  $P_x^<$ ,  $P_x^{\leq}$  and  $P_x^=$  can all be defined in terms of the basic one (having self-explanatory meanings). In addition,  $P_1^{\geq}$  can be identified with the classical (KD45) modal belief operator  $B$ . An important feature of  $P_FKD45$  is that, although its syntax allows for an operator  $P_x^>$  for every rational number  $x$  in the interval  $[0, 1]$ , in the semantics it is assumed that probabilities are only taken from a finite base  $F = \{r_0, r_1, \dots, r_n\} \subseteq [0, 1] \cap \mathbb{Q}$  thus allowing representation of probabilities in terms of this simple set.

This restriction first of all restores *compactness* of the logic (without the finiteness constraint on  $F$ , note that  $\Gamma = \{P_\alpha^>q \mid \alpha \in [0, 1]\}$  *semantically entails* that  $P_1^=q$ , but no finite subset of  $\Gamma$  can *prove*  $P_1^=q$ ). Related to this, when searching for models for formulae of  $P_FKD45$ , the constraint on  $F$  implies that we only have to consider finitely many different models. In addition, it appears that the restriction to a finite base,  $F$ , still allows us to model realistic problems: agents in general do not need an infinite granularity to express their uncertainty. For example, if the probabilistic belief in some fact is under a certain threshold, the agent might either assume the probability is 0, or, in case the certainty about  $\varphi$  has become too low to act upon, it might want to carry out a sensing operation to increase its confidence in  $\varphi$ .

In short,  $P_FKD45$  builds upon the natural framework of Kripke models (the basis of modal logics), while extending it to a probability structure. Formulae are interpreted on what are called *Probabilistic Kripke Models over  $F$*  (or  $\mathcal{P}_FKD45$  models). That is,  $P_x^>\varphi$  is true at a world  $w$  if, and only if, the probability values in  $F$  that are assigned to the possible worlds which verify  $\varphi$ , sum up to a value greater than  $x$ . Note that these probabilities are assigned globally, and do not depend on the world  $w$  of evaluation. Properties of  $P_FKD45$  include soundness, completeness and finite modelling for consistent formulae [8]. Another property is that *nested* belief formulae can be removed, i.e. any nested belief formula is equivalent to one without nesting. In [8] we described a decision procedure for  $P_FKD45$ , which we will call  $P_FKD45 \text{ DEC PROC}(\cdot)$  in this paper. This decision procedure aims at finding a finite model for an agent specification ( $\varphi$ ), if such a model exists. Roughly,  $P_FKD45 \text{ DEC PROC}(\cdot)$  first transforms  $\varphi$  into a normal form, without nesting of probabilistic operators, and then translates, for instance  $P_{0.8}^{\geq}p \wedge P_{0.7}^{\geq}q$ , formulae into a system of linear inequalities<sup>2</sup>:

$$\begin{aligned} p0q0 + p0q1 + p1q0 + p1q1 &= 1 \\ p1q0 + p1q1 &\geq 0.8 \\ p0q1 + p1q1 &\geq 0.7 \end{aligned}$$

Inequalities, together with the constraints  $piqj \in F$ , are fed to a constraint solver, which generates a possible solution,  $P_F$ , if this exists. Note that we can conceive of every combination,  $piqj$ , as a set of possible worlds, or, if  $p$  and  $q$  are the only propositional atoms, as possible worlds<sup>3</sup>.

### 2.3 Combining Time and Probabilistic Belief

We now combine the METATEM and  $P_FKD45$  approaches to derive PROTEM. By a *fusion* of the two logics we guarantee [15] that several key properties of both of these

<sup>2</sup> Disjunctions of (sets of) formulae are attempted one at a time, as explained in Section 4.1.

<sup>3</sup> In such a case,  $p0q1$  for instance indicates the assigned probability to the set of worlds satisfying  $\neg p \wedge q$ .

logics are preserved in PROTEM. The language of PROTEM is obtained by the union of the two underlying languages, where we assume the basic operators to be **start**,  $\bigcirc$ , and  $\mathcal{U}$  for the temporal component, and  $P_x^>$  for the probabilistic component. Other operators, such as  $\square$ ,  $\diamond$ ,  $\mathcal{W}$  (weak until) and  $P_x^\sim$  (where  $\sim$  is in  $\{\geq, =, \leq, <\}$ ) can be defined in terms of the basic ones.

A model for PROTEM,  $M$ , can be conceived of as having countably many timelines  $\ell, \ell', \ell'', \dots$ , each being a copy of  $\mathbb{N}$ . A world in  $M$  is a pair  $(\ell, i)$  with  $i \in \mathbb{N}$ . We assume there is a probability function  $P_i$  for every  $i$ , such that  $P_i(\{(\ell, j) \mid j = i\}) = 1$ , and for every  $\ell$ ,  $P_i(\ell, i) \in F$ , the finite based set for the probabilities. Below we only give the main clauses for the truth definition.

$$\begin{aligned} M, (\ell, i) &\models \mathbf{start} \text{ iff } i = 0 \\ M, (\ell, i) &\models \bigcirc\varphi \text{ iff } M, (\ell, i + 1) \models \varphi \\ M, (\ell, i) &\models \varphi\mathcal{U}\psi \text{ iff } \exists j [(j > i) \ \& \ M, (\ell, j) \models \psi \\ &\quad \& \ \forall k (i \leq k < j \Rightarrow M, (\ell, k) \models \varphi)] \\ M, (\ell, i) &\models P_x^>\varphi \text{ iff } P_i(\{(\ell', i) \mid M, (\ell', i) \models \varphi\}) > x \end{aligned}$$

Note that the truth valuation of  $P_x^>\varphi$ -formulae does not depend on the specific timeline  $\ell$ :  $M, (\ell, i) \models P_x^>\varphi$  iff  $M, (\ell', i) \models P_x^>\varphi$ .

Axioms of the language, as shown in Fig. 1, reflect basic properties of probability theory, together with the peculiarity of having this base set  $F$  (axiom A10). Those axioms can also be found in [8], together with some meta-theorems about  $P_FKD45$ .

Since the two underlying logics are well behaved [3,8] and fusion, in general, preserves good behaviour [15] we obtain the following.

**Theorem 1.** *The logic of PROTEM:*

- \* is sound and complete with respect to the above semantics;
- \* is decidable; and
- \* has the finite model property.

A1 All propositional tautologies	A2 $\bigcirc\neg\varphi \Leftrightarrow \neg\bigcirc\varphi$
A3 $\bigcirc(\varphi \Rightarrow \psi) \Rightarrow (\bigcirc\varphi \Rightarrow \bigcirc\psi)$	A4 $\varphi\mathcal{U}\psi \Leftrightarrow (\psi \vee (\varphi \wedge \bigcirc(\varphi\mathcal{U}\psi)))$
A5 $P_1^\geq(\varphi \Rightarrow \psi) \Rightarrow [(P_x^\sim\varphi \Rightarrow P_x^\sim\psi)]$	A5' $P_1^\geq(\varphi \Rightarrow \psi) \Rightarrow [(P_x^>\varphi \Rightarrow P_x^\geq\psi)]$
A6 $P_1^\geq(\varphi \Rightarrow \psi) \Rightarrow (P_x^\geq\varphi \Rightarrow P_x^\geq\psi)$	A7 $P_0^\geq\varphi$
A8 $P_{x+y}^\geq(\varphi \vee \psi) \Rightarrow (P_x^\geq\varphi \vee P_y^\geq\psi)$ $\Rightarrow ((P_x^\geq\varphi \wedge P_y^\geq\psi) \Rightarrow P_{x+y}^\geq(\varphi \vee \psi))$	A9 $P_1^\geq\neg(\varphi \wedge \psi)$
A10 $P_{r_i}^\geq\varphi \Rightarrow P_{r_{i+1}}^\geq\varphi$	A11 $(P_0^\geq P_x^\geq\varphi \Rightarrow P_x^\geq\varphi) \wedge (P_0^\leq P_x^\leq\varphi \Rightarrow P_x^\leq\varphi)$
A12 $(P_x^\geq\varphi \Rightarrow P_1^\geq P_x^\geq\varphi) \wedge (P_x^\leq\varphi \Rightarrow P_1^\geq P_x^\leq\varphi)$	A13 $\square(\varphi \Rightarrow \bigcirc\varphi) \Rightarrow (\varphi \Rightarrow \square\varphi)$
R1 $\vdash\varphi, \vdash\varphi \Rightarrow \psi \rightarrow \vdash\psi$	
R2 $\vdash\varphi \rightarrow \vdash\bigcirc\varphi$	R3 $\vdash\varphi \rightarrow \vdash P_1^\geq\varphi$

**Fig. 1.** Axioms of PROTEM. Everywhere,  $x, y, z, x + y \in [0, 1] \cap \mathbb{Q}$ , and  $z < x$ . In A10, the numbers  $r_i$  and  $r_{i+1}$  are both in  $F$ . In A5,  $\sim \in \{>, \geq\}$ .

### 3 Normal Form for PROTEM

We will now describe a transformation,  $\tau$ , that takes an arbitrary formula from PROTEM and returns a formula of the form  $\Box^* \wedge F_i$  where each  $F_i$  is a formula in one of the following four forms.

$$\begin{aligned}
 \mathbf{start} &\Rightarrow \bigvee l_b \\
 \bigwedge k_a &\Rightarrow \bigcirc \bigvee l_b \\
 \bigwedge k_a &\Rightarrow \diamond l \\
 \bigwedge k_a &\Rightarrow \bigvee P_x^\sim l_b
 \end{aligned} \tag{1}$$

Here  $k_a, l_b$  and  $l$  are classical literals,  $\sim$  is in  $\{\geq, =, \leq, <\}$  and  $\Box^*$  represents the universal modality. This normal form is an extension of that (called SNF) developed both for temporal logics and their modal extensions [13].

In this way, we split the satisfiability problem into a part relating to the beginning of time (**start**), a part that relates to the next state, a part that collects the eventualities, and a part that deals with the probabilistic beliefs (at the current time).

The transformation  $\tau$  works as follows, on a given PROTEM formula  $\varphi$ . First of all, it yields (where  $f$  is a new atom)  $\mathbf{start} \Rightarrow f, f \Leftrightarrow \varphi$ . Next, complex subformulae are renamed, for example  $\diamond(\psi \wedge \varphi)$  becomes  $\diamond g, g \Leftrightarrow (\psi \wedge \varphi)$ , where  $g$  is a new atom. For any  $h \Leftrightarrow \psi$  that is generated ( $h$  an atom,  $\psi$  not already in normal form),  $\tau$  replaces  $h \Leftrightarrow \psi$  by  $h \Rightarrow \psi, \neg h \Rightarrow \neg\psi$ . In the temporal case, more complex operators such as ' $\Box$ ' are reduced to their fixpoint definitions in terms of ' $\bigcirc$ ' and ' $\diamond$ '. The reader might compare this with [13,11], to which the ' $\Leftrightarrow$ ' cases were added.

For the treatment of  $P_x^\sim$  formulae, we carry out the following. First of all, we know that such formulae can all be rewritten using only  $P_y^>$  operators. Secondly, by [8–Theorem 2], we may assume that  $\varphi$  is without any direct nesting of operators  $P_x^>$ . That is, nested probabilistic operators collapse to a single non-nested one. So, if any  $P_x^>$  occurs in the scope of a  $P_y^>$ , there must be a temporal operator that separates them. For  $\tau$ , we do the following: replace all occurrences of  $h \Rightarrow D \vee P_x^>\psi$  (where  $D$  is a disjunction and  $\psi$  is not a literal) by  $h \Rightarrow D \vee P_x^>g, g \Leftrightarrow \psi$  (where  $g$  is a fresh atom). Similarly, occurrences of  $h \Rightarrow D \vee \neg P_x^>\psi$  are rewritten to  $h \Rightarrow D \vee \neg P_x^>g, g \Leftrightarrow \psi$ .

**Theorem 2.** *If a PROTEM formula,  $\varphi$ , is satisfiable, then so is  $\tau(\varphi)$ .*

Since  $\tau$  introduces new atoms, we are not *a priori* guaranteed, when building a model for  $\tau(\varphi)$ , to have a model for  $\varphi$ . However, using ' $\Leftrightarrow$ ' to define new atoms in  $\tau$  takes care of this:

**Theorem 3.** *If  $\tau(\varphi)$  is satisfied in  $M$ , then so is  $\varphi$ .*

As presented in [13], the use of ' $\Leftrightarrow$ ' in the renaming transformation of formulae implies a potential exponential increase in terms of the size of the set of formulae. This is due to the fact that, in the worst case, all the subformulae of the initial formulae would have to be replaced by new propositional symbols. However, it is important to notice that in practise the complexity of this renaming tends to be much smaller.

## 4 Execution Mechanism

Now we consider execution of the formulae within the above normal form. The semantics for the temporal component is based on models that are discrete linear orders. From [8], we know that models for the probabilistic part of our language can be conceived of as structures  $\langle W, w, P_F \rangle$ , where  $W$  is a set of valuations,  $w$  is a designated world and  $P_F : W \rightarrow F$  is a probability function, yielding a probability value from  $F$  to every valuation. These models are the probabilistic version of ‘ $KD45$ -balloons’: every balloon represents valuations (or possible  $w$ ). If  $w$  is not doxastically possible, it receives a probability of 0. Otherwise, it receives a positive probability value. For simplicity, we will refer to such  $P_FKD45$  models as ‘balloons’. An easy calculation gives an upper bound on the number of possible different models for a formula  $P_FKD45$ -formula  $\varphi$ :

**Lemma 1.** *Suppose  $\varphi$  has  $n = |\varphi|$  atoms. Then  $\varphi$  is  $P_FKD45$  satisfiable if, and only if,  $\varphi$  has a model  $\langle W, w, P_F \rangle$  with  $|W \cup \{w\}| \leq 2^n$ . Moreover, if  $F$  consists of  $|F|$  different numbers, there are at most  $|F|^{2^n}$  different models for a formula in  $n$  atoms.*

Note that the number of different models for  $\varphi$  is given by a rough upper bound: it considers *all* possible assignments of probabilistic values from  $F$  to conjunctions of objective literals, whereas in practise many of them will not be real probability assignments (with the property that the sum of any two partitions of such conjunctions must be 1, for example).

Adding this result on the upper bound for models for the temporal component, we obtain<sup>4</sup>:

**Theorem 4.** *Let  $n = |\varphi|$ . Then  $\varphi$  is satisfiable in PROTEM if, and only if, it is satisfiable in a model with at most  $2^{5 \cdot n}$  balloon models (in the temporal dimension). Moreover, we only need to consider at most  $|F|^{2^n}$  different balloon models (in the probabilistic dimension).*

*Remark 2.* It is important to note that it is generally only in the case of unsatisfiable specifications that *all* these possibilities must be explored. Typically, the search for an execution does not consider all these models.

The key idea underlying METATEM is to directly execute a temporal formula by attempting to build a model (as a sequence of states) for the agent description (a single logical statement has both declarative and procedural readings). When including beliefs, instead of generating a set of choices based upon temporal rules, both temporal and belief rules must be considered. That is, the execution now uses temporal formulae in the normal form to build a temporal sequence of *balloons*. When no temporal rules are fired, we have to ‘fill a balloon’ and explore the beliefs that currently should be satisfied. For this, we use the decision procedure of [8], which, in turn, heavily relies on a constraint solver for linear inequalities (generated by the probabilistic formulae). Of course, if the solver finds a solution, it means that in the newly generated balloons

<sup>4</sup> The simulation of temporal states is, by definition, infinite. However, as we will explain in Section 4.2, there is an upper bound for the number of unsatisfiable temporal states generated in our execution. And this leads us to the result in Theorem 4.

we may have some atoms that give rise to new potential timelines in the belief context (see Fig. 2, keeping in mind that every bullet ‘•’ represents a balloon  $\langle W, w, P_F \rangle$ ). The execution mechanism is organised in such a way that this exploration of beliefs will not continue forever.

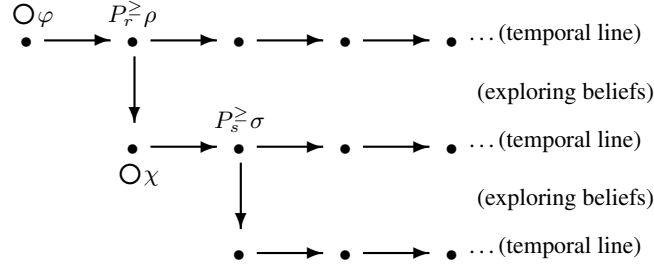


Fig. 2. Sample PROTEM Model Construction

We claim that this is a complete algorithm that can, if necessary, be used as a decision procedure for the combined logic. We know that the extended METATEM +  $P_FKD45$  framework, PROTEM, inherits the finite modelling properties that both systems have. We further show that the execution algorithm is effective in searching for a specification’s model: it either finds a model for the specification or ends the execution failing to produce one (having exhaustively explored all the possible execution paths). In the case of failure, we know that the specification is unsatisfiable.

#### 4.1 The Algorithm

In this section we describe an execution algorithm that constructs a model for a formula  $\varphi$  in the normal form. We will refer to the implications as displayed in (1) as the set of rules,  $R$ , of  $\varphi$ , and attempt to build a model for  $R$ , if one exists. The model construction (hence, execution) algorithm basically comprises two mutually recursive procedures, TEMPEXPAND and PROBEXPAND.

The function TEMPEXPAND( $R, s, Ev$ ) attempts to build a temporal model from the (belief) state,  $s$ . It follows the previous METATEM algorithm presented in [4] and so we only give an informal outline here. At any given state, the algorithm checks which rules (within  $R$ ) are relevant and collects together the constraints on future temporal execution. These constraints are represented as a disjunction, together with a number of eventualities yet to be satisfied. During execution, a disjunct is chosen and as many eventualities are satisfied as possible. The disjunct, together with the satisfied eventualities, generate a new temporal state which will be used in a recursive call of TEMPEXPAND; similarly, unsatisfied eventualities are passed on to this recursive call. In this way, a temporal sequence is constructed. If backtracking occurs, then an alternative disjunct is chosen. If no unexplored alternative exists, then the rules  $R$  represent an unsatisfiable formula.

At each state,  $s$ , being constructed using TEMPEXPAND, the relevant *probabilistic* constraints are examined using PROBEXPAND, which carries out probabilistic belief

exploration. Given the current state  $s$ , it looks for all disjunctions of probabilistic literals that need to be satisfied in  $s$ . If there are no such disjunctions, we successfully terminate and return **true**. Otherwise, to make all disjunctions true in  $s$ , we have to satisfy at least one combination of disjuncts  $d_i$ , one for each  $D_i = d_{1_i} \vee \dots \vee d_{n_i}$ . We collect all those possible combinations as conjunctions in Unexplored Options (UnexplOpt). Now,  $\text{PROBEXPAND}(R, s)$  should return **true** if, and only if, UnexplOpt is satisfiable.

For each such combination  $C$  in UnexplOpt, we call the decision procedure for  $P_FKD45$ . If  $C$  is satisfiable, this procedure generates a set of possible  $P_FKD45$  models, which we call “Balloons” in the procedure. Each of these Balloons,  $t$ , is then exposed to  $\text{TEMPEXPAND}$ , which tries to solve the temporal constraints for  $t$ . If it succeeds, we are done. Otherwise we choose another Balloon  $t'$ . If none yields a successful temporal expansion, we try another possibilistic combination  $C'$  from UnexplOpt. If all possibilities fail, then there are no satisfiable possibilities at this point and backtracking occurs.

```

PROBEXPAND( $P, R, s$ )
1  ProbDis  $\leftarrow \{\vee P_x \tilde{g} \mid h \Rightarrow \vee P_x F \in R \text{ and } s \models h\}$ 
2  if ProbDis =  $\emptyset$  then
3    return true
4  endif
5  UnexplOpt  $\leftarrow \emptyset$ 
6  for  $D_i \leftarrow D_1 = \vee P_x \tilde{g}, D_2, \dots, D_n \in \text{ProbDis}$  do
7    select exactly one disjunct  $d_i$  from  $D_i$ 
8    UnexplOpt  $\leftarrow \text{UnexplOpt} \cup \{d_1 \wedge \dots \wedge d_n\}$ 
9  endfor
10 while UnexplOpt  $\neq \emptyset$  do
11   Select  $C$  from UnexplOpt
12   Balloons =  $\{\langle W, w, P_F \rangle \leftarrow P_FKD45 \text{ DEC PROC}(C)\}$ 
13   while Balloons  $\neq \emptyset$  do
14     select  $t = \langle W, w, P_F \rangle$  from Balloons
15     if  $\text{TEMPEXPAND}(R, t, \emptyset)$  succeeds then
16       return true
17     endif
18     Balloons  $\leftarrow \text{Balloons} \setminus \{t\}$ 
19   endwhile
20   UnexplOpt  $\leftarrow \text{UnexplOpt} \setminus \{C\}$ 
21 endwhile
22 return false

```

**Fig. 3.** Procedure  $\text{PROBEXPAND}$

## 4.2 Correctness of Execution Algorithm

As to the correctness of  $\text{METATEM}$ , [4] states that a set of formulae,  $R$ , is satisfiable, if, and only if, the  $\text{METATEM}$  interpreter generates a model for  $R$  (here, we will only

require a boolean if the model exists). The ‘oldest eventualities’ are attempted first at each step; when the number of attempts to satisfy any eventuality reaches  $2^{5|R|}$ , backtracking is forced and another choice is attempted.

The proof for the whole extended system follows by showing that the execution with probabilistic beliefs (potentially) explores all the linear sequences of states. In detail, the execution algorithm implements a search in the space of possible models for the given beliefs, together with a simulation of temporal aspects associated with them. Thus, the search explores models in two dimensions: a doxastic and a temporal one. In terms of beliefs, we know that the search represents a simple verification of a probability assignment that satisfies the given formulae. By the finite model property of the language, we know that every consistent set of formulae has a finite model. And the  $P_FKD45$  decision procedure is used to obtain a model for the set of formulae, or return an empty set of probability assignments if the set is unsatisfiable. The second dimension is the temporal line to be simulated. This is done by using a construction of temporal states similar to the basic algorithm for the METATEM framework.

**Lemma 2.** *Given a set of formulae,  $R$ , and a state,  $s$ , **PROBEXPAND** returns **false** iff  $R$  has no model.*

As a proof outline, recalling how the computation tree is built, we know that the algorithm starts by exploring one of the disjuncts of the given formula. We also know that **PROBEXPAND** has a finite number of potential ways to recurse. Recall that **PROBEXPAND** invokes the  $P_FKD45$  decision procedure in order to verify the existence of a model for the given set of formulae. So, if in a node the given formulae are unsatisfiable, no model is going to be generated. If all branches express unsatisfiable formulae, all the possible disjunctions would be unsatisfiable and the algorithm would fail to produce a model for the graded beliefs.

**Theorem 5.** *A set of rules,  $R$ , is satisfiable if, and only if, the extended framework algorithm returns **true**.*

*Remark 3.* Note that part of the exploration of probabilistic beliefs involves also a simulation of temporal aspects, by calling **TEMPEXPAND**. This part of the algorithm is similar to the basic **METATEM** one. Moreover, one can extend the two procedures in such a way that they generate the required model, rather than returning **true**, if it exists.

## 5 Conclusion

In previous research on executable agent specifications, the problem of handling uncertainty has rarely been tackled, an exception being extensions of Logic Programming providing *probabilistic logic programs* [9]. In contrast, our approach uses non-classical logics as the basis for agent specification and forward chaining as the basis for agent execution.

Simple and expressive formalisms that deal with the combination of uncertainty and time have many interesting and non-trivial applications. For instance, *Probabilistic Planning* [7] is a field where both probabilistic beliefs and temporal aspects are crucial.

In Computer Science, reasoning about probabilistic properties of concurrent systems is increasingly important [1]. Also, recently, reasoning about agents in game-like scenario has received considerable attention from the Logic community [2]. Typically, reasoning about strategies in such games has a temporal aspect and, moreover, games with *incomplete information* are only recently put fully on the logical agenda. Moreover, for certain games, it is well-known that only if we allow agents to play *mixed strategies*, which means that they play each (*pure*) strategy with a certain probability, solutions like that of Nash Equilibrium are guaranteed (c.f. [18]). In these, and other, areas we believe that PROTEM offers an appropriate framework in which to allow agents to reason about uncertain information.

There are many directions to further our framework. First of all, PROTEM represents the simplest way to combine two logics, i.e. via a fusion. When adding additional assumptions on the relation between the temporal and the doxastic dimensions, it is not clear whether we can adjust our algorithm easily. And, if so, will we retain important results? The extension to *multi-agent* systems seems less problematic, from this point of view, since the most straightforward way to extend PROTEM to the multi-agent case still is a fusion of several logics. It becomes more intricate when adding even the simplest dependencies, however, such as the property that what is believed by a given agent *A*, is also believed by another agent *B*.

## Acknowledgement

The first author gratefully acknowledges support by the Brazilian Government under CAPES-scholarship.

## References

1. P. Abdulla, C. Baier, P. Iyer, and B. Jonsson. Reasoning about Probabilistic Lossy Channel Systems *LNCS*, 1877, 2000.
2. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, **49**(5): 672 – 713, 2002.
3. H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: An Introduction. *Formal Aspects of Computing*, **7**(5): 533 – 549, 1995.
4. H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logics*. Research Studies Press, Chichester, UK, 1996.
5. J. Bradshaw, M. Greaves, H. Holmback, T. Karygiannis, B. Silverman, N. Suri, and A. Wong. Agents for the Masses? *IEEE Intelligent Systems*, **14**(2), 1999.
6. P. R. Cohen and H. J. Levesque. Intention Is Choice with Commitment. *Artificial Intelligence*, **42**(2-3): 213 – 261, March 1990.
7. International Planning Competition: Probabilistic Planning Track., 2004. <http://www.cs.rutgers.edu/~mlittman/topics/ipc04-pt>.
8. N. de Carvalho Ferreira, M. Fisher, and W. van der Hoek. Practical Reasoning for Uncertain Agents. In *Proc. Ninth European Conf. on Logics in Artificial Intelligence (JELIA)*, pp. 82 – 94, 2004.
9. J. Dix, S. Kraus, and V. S. Subrahmanian. Heterogenous Temporal Probabilistic Agents. *ACM Transactions on Computational Logic*, **5**(3), 2004.

10. M. Fisher and C. Ghidini. Programming Resource-Bounded Deliberative Agents. In *Proc. International Joint Conf. on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 1999.
11. M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, **2**(1): 12 – 56, January 2001.
12. M. Fisher. Representing and Executing Agent-Based Systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents*. Springer-Verlag, 1995.
13. M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *J. Logic and Computation*, **7**(4): 429 – 456, 1997.
14. J. Y. Halpern. *Reasoning About Uncertainty*. MIT Press, 2003.
15. M. Kracht and F. Wolter. Properties of Independently Axiomatizable Bimodal Logics. *Journal of Symbolic Logic*, **56**(4):1469 – 1485, 1991.
16. P. Madhusudan and P.S.Thiagarajan. Branching-time Controllers for Discrete Event Systems. *Theoretical Computer Science*, **274**: 117 – 149, 2002.
17. Z. Manna and P. Wolper. Synthesis of Communication Processes for Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, **6**: 68 – 93, 1984.
18. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
19. A. S. Rao and M. Georgeff. BDI Agents: from Theory to Practice. In *Proc. First International Conf. on Multi-Agent Systems (ICMAS)*, pp. 312 – 319, 1995.
20. W. van der Hoek and M.J. Wooldridge. Towards a Logic of Rational Agency. *Logic Journal of the IGPL*, **11**(2): 135 – 160, 2003.
21. M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, 2000.