Agents dealing with Time and Uncertainty

Jürgen Dix*
The University of Manchester
Dept. of CS, Oxford Road
Manchester M13 9PL, UK
dix@cs.man.ac.uk

Sarit Kraus†
Dept. of CS/UMIACS
Bar-Ilan/College Park
Israel/USA
sarit@cs.macs.biu.ac.il

VS Subrahmanian‡
University of Maryland
Dept. of CS
College Park, MD 20752, USA
vs@cs.umd.edu

ABSTRACT

Situated agents in the real world need to handle the fact that events occur frequently, as well as the fact that the agent typically has uncertain knowledge about what is true in the world. The ability to reason about both time and uncertainty is therefore very important. In this paper, we develop a formal theory of agents that can reason about *both* time and uncertainty. The theory extends the notion of agents described in [10, 21] and proposes the notion of *temporal probabilistic (or* TP*) agents*. A formal semantics for TP-agents is proposed - this semantics is described via structures called *feasible* TP*-status interpretations* (FTPSI's). TP-agents continuously evaluate changes (in the state of the environment they are situated in) and compute appropriate FTPSI's. For a class of TP-agents called *positive* TP-agents, we develop a provably sound and complete procedure to compute FTPSI's.

Categories and Subject Descriptors

I.2.12 [**Artificial Intelligence**]: Distributed AI—*Intelligent Agents*

Keywords

Formalisms and logics: logic programming, Theories of agency Probabilistic/uncertain reasoning, Temporal reasoning

1. INTRODUCTION

There are numerous applications where agents situated in the real world need to reason about both time and uncertainty. For example, an agent monitoring temperature readings at a manufacturing plant may need to make predictions about future expected temperatures based on regression techniques. This agent may need to take different actions based on when the temperature is expected to reach a certain value and with what probability. Likewise, an agent built on top of a legacy stock market model that predicts what stock prices might be in he future, may wish to execute trades at the "right" time based on the some appropriate temporal-probabilistic analysis.

It is well known that logics to reason simultaneously about *both* time and probabilities are complex [11, 12]. Hence methods to build agents that not only reason about time and probabilities, but also take actions based on such reasoning, is even more complex. A further complication arises when such agents are built on top of legacy pieces of software code that already exist. For example, building agents that decide what trades to make and when to make them in the stock market example referenced above can be quite complex.

In this paper, we develop the concept of a temporal probabilistic (TP) agent which is built on top of legacy software code or legacy data sources. Methods to build agents on top of legacy code have been studied in [21]. The *state* of an agent at a given point in time is the set of all objects residing in the data structures of the code on top of which the agent is built. Our primary technical contributions are the following: first, we develop a formal syntax for TP-agent programs and then provide a formal declarative semantics for such TP-agents. The declarative semantics is given in terms of certain structures called *feasible temporal probabilistic status interpretations* (FTPSI for short). Agents continuously engage in the following computational cycle: identify changes to state → compute FTPSI → take actions prescribed by FTPSI → identify changes to state → .... Thus, the computation of FTPSI's is a continuous ongoing process. We present an algorithm to compute FTPSI's. This algorithm is provably sound and complete for a large class of TP agents called *positive* TP agents.

1.1 Motivating (Stock) Example

Consider a very simple agent that tracks stocks and executes trades automatically for clients. A specific client may wish to trade stocks based on some simple rules. For example, if a prediction program predicts that a given stock is going to go up to $50 per share with a high probability (e.g. 80% or more) sometime during the next 10-20 days

*Responsible Author. The author gratefully acknowledges support from EPSRC grant GR/R57843/01 *Optimizations of MAS*.

†The author gratefully acknowledges support from NSF grant # IIS9907482.

‡This work was supported by the Army Research Laboratory under contract number DAAD190010484 and by an NSF Young Investigator award.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

and the user already owns this stock, then she may want to buy the stock sometime in the next 9 days. However, if the user does not own this stock (and if it is consistent with the user's diversified investment strategy) then the user may want a 90% (or more) probability that the stock will go up to $50 in order to buy it.

Throughout this paper, we will use this simple example to motivate the need for taking actions automatically during the presence of temporal uncertainty. As will become clear, our notion of temporal probabilistic agents can be easily used to write far more sophisticated agents with more complex rules than those described above. However, due to space constraints, we are unable to do so here.

2. TEMPORAL PROBABILISTIC CC'S

To perform logical reasoning on top of third party data structures (which are part of the agent's state) and code, the agent must have a language within which it can reason about the agent state. In [10], we introduced the concept of a *code call*, which is the basic syntactic object used to access multiple heterogeneous data sources. In this section, we are extending this notion to incorporate time and uncertainty: temporal probabilistic code call (TPCC).

Definition 2.1 (Code Call (*cc*)) *Suppose* $\mathcal{S} =_{def} (\mathcal{T_S}, \mathcal{F_S})$ *is some software code,* $f \in \mathcal{F_S}$ *is a predefined function with n arguments, and* $\mathtt{d_1}, \ldots, \mathtt{d_n}$ *are objects or variables such that each* $\mathtt{d_i}$ *respects the type requirements of the i'th argument of f. Then,* $\mathcal{S}:f(\mathtt{d_1}, \ldots, \mathtt{d_n})$ *is a* code call. *A code call is* ground *if all the* $\mathtt{d_i}$*'s are objects.*

We often identify software code $\mathcal{S}$ *with the agent that is built by it.*

A code call executes an *API* function and returns as output a set of objects of the appropriate output type.

The situation changes drastically when there is uncertainty about the state. In this case, we cannot assume that a code call returns a complete answer — rather, it may return a set of random variables [6].

Definition 2.2 (Coherent set of RV's of type τ) *A* random variable **RV** *of type* τ *is a pair* $\mathbf{RV} = (Obj, \wp)$ *where Obj is a finite set of elements of type* τ *and* $\wp$ *is a probability distribution over Obj that assigns real numbers in the unit interval* $[0, 1]$ *to members of* **RV** *such that* $\Sigma_{o \in Obj} \wp(o) \leq 1$.

A set S of random variables of type τ *is* coherent *iff whenever we consider two distinct* $(Obj_1, \wp_1), (Obj_2, \wp_2) \in S$*, it is the case that* $Obj_1 \cap Obj_2 = \emptyset$.

This allows us to introduce the concept of a TP code call.

Definition 2.3 (Temporal probabilistic *cc* (TPcc)) *A* temporal probabilistic code call based on *cc* , *denoted* cc^{TP}*, returns a* mapping *from natural numbers to coherent sets of random variables of type* τ.

Suppose we have a code call $\mathtt{stock}:over(\mathtt{C}, \mathtt{P})$ which ordinarily returns as output, a member of $\{\mathbf{true}, \mathbf{false}\}$ indicating whether the company C's stock price is P or more. A TPCC based on this code call would return a mapping as output. This mapping would associate with each time point t, a set of random variables. For example, at time 1 it may return only one random variable, $(\{\mathbf{true}, \mathbf{false}\}, \delta_u)$ saying that at time 1 there is a 50% probability of the stock being greater than or equal to P (i.e., δ_u is the uniform distribution function). At time 2, it may return the same set, but with the distribution $\wp(\mathbf{true}) = 0.8, \wp(\mathbf{false}) = 0.2$ indicating that there is now an 80% probability of C's stock exceeding $50. In the rest of this paper, we will often write *cc* to both denote ordinary code calls as well as TPCC's. The intended meaning should be clear from context.

It is well known that even if we know the exact probabilities of two events e_1, e_2, it is not always easy to obtain a precise point probability for the conjunction or disjunction of these events, though it is possible to obtain a "tightest" possible interval for these probabilities (cf. Boole[2], Fagin et. al.[12]). In general, depending upon exactly what is known about the two events, the probabilities of their conjunction and disjunction can be computed in many different ways. The following definition (due to Lakshmanan et. al. [18]) proposes the notion of a probabilistic conjunction/disjunction strategy which provides an abstract view of how to obtain probabilities of conjunctive/disjunctive events.

Given two probability intervals $[L_1, U_1]$ and $[L_2, U_2]$, we say that $[L_1, U_1] \leq [L_2, U_2]$ *iff* $L_1 \leq L_2$ and $U_1 \leq U_2$.

Definition 2.4 (Probabilistic conj/disj strategy) *Let events* e_1, e_2 *be associated probabilistic intervals* $[L_1, U_1]$ *and* $[L_2, U_2]$ *respectively. Then a* probabilistic conjunction strategy *(*probabilistic disjunction strategy*) is a binary operation* $\otimes$ *(*$\oplus$*) which uses this information to compute the probabilistic interval* $[L, U]$ *for event "*$e_1 \wedge e_2$*" ("*$e_1 \vee e_2$*"). When the events involved are clear from context, we use* $[L, U] = [L_1, U_1] \otimes [L_2, U_2]$ *to denote* $(e_1 \wedge e_2, [L, U]) = (e_1, [L_1, U_1]) \otimes (e_2, [L_2, U_2])$ *and we use* $[L, U] = [L_1, U_1] \oplus [L_2, U_2]$ *to denote* $(e_1 \vee e_2, [L, U]) = (e_1, [L_1, U_1]) \oplus (e_2, [L_2, U_2])$. *Every strategy must conform to the following postulates:*

Generic postulates $(* \in \{\otimes, \oplus\})$	
1. Commutativity	$([L_1, U_1] * [L_2, U_2]) = ([L_2, U_2] * [L_1, U_1])$
2. Associativity	$(([L_1, U_1] * [L_2, U_2]) * [L_3, U_3]) =$ $([L_1, U_1] * ([L_2, U_2]) * [L_3, U_3]))$
3. Monotonicity	$([L_1, U_1] * [L_2, U_2]) \leq ([L_1, U_1] * [L_3, U_3])$ *if* $[L_2, U_2] \leq [L_3, U_3]$
	Probabilistic Conjunction postulates
4.a. Bottomline	$([L_1, U_1] \otimes [L_2, U_2])$ $\leq [\min(L_1, L_2), \min(U_1, U_2)]$
5.a. Identity	$([L_1, U_1] \otimes [1, 1]) = [L_1, U_1]$
6.a. Annihilator	$([L_1, U_1] \otimes [0, 0]) = [0, 0]$
7.a. Ignorance	$([L_1, U_1] \otimes [L_2, U_2]) \subseteq$ $[\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
	Probabilistic Disjunction postulates
4.b. Bottomline	$([L_1, U_1] \oplus [L_2, U_2]) \geq$ $[\max(L_1, L_2), \max(U_1, U_2)]$
5.b. Identity	$([L_1, U_1] \oplus [0, 0]) = [L_1, U_1]$
6.b. Annihilator	$([L_1, U_1] \oplus [1, 1]) = [1, 1]$
7.b. Ignorance	$([L_1, U_1] \oplus [L_2, U_2]) \subseteq$ $[\max(L_1, L_2), \min(1, U_1 + U_2)]$

A detailed explanation of why these axioms are reasonable axioms for probabilistic reasoning is given in [18]. Due to space reasons we are unable to go into it here.

Each agent has a set of actions that the agent is capable of executing. Actions change the state of the agent. An action has five components: (1) a name $\alpha(X_1, ..., X_n)$ where the $\mathtt{X}_i$'s are variables, (2) a *precondition*, (3) an add list, (4) a delete list, all three of which consist of a set of code calls and temporal probabilistic code calls, and (5) an action *code* which is a body of code that executes the action. Each

agent has a notion of *concurrency* specifying how to combine a set of actions into a single action, and a set of *action constraints* that define the circumstances under which certain actions may be concurrently executed. A formal model of such agents (with no time and no uncertainty) is given in [10]. For simplicity, we assume (in this paper) that an action has no duration (see [5] for how to handle actions with durations).

Definition 2.5 (Status conjunction, Status set) *If $\alpha(\vec{t})$ is an action, and* Op *belongs to* $\{\mathbf{P}, \mathbf{F}, \mathbf{W}, \mathbf{Do}, \mathbf{O}\}$, *then* $\mathsf{Op}\alpha(\vec{t})$ *is called a* status atom. *If $A_1, \ldots, A_n$ are status atoms, then $(A_1 \wedge \ldots \wedge A_n)$ is a* status conjunction. *A* status set *is a finite set of ground status atoms.*

Intuitively, $\mathbf{P}\alpha$ means α is permitted. $\mathbf{F}\alpha$ means α is forbidden. $\mathbf{Do}\,\alpha$ means α is actually done. $\mathbf{O}\alpha$ means α is obliged, and $\mathbf{W}\alpha$ means that the obligation to perform α is waived.[1]

3. SYNTAX OF TP AGENT PROGRAMS

Our main aim in this section is to define *temporal probabilistic* agents. A first step is to extend ccc's and status conjunctions (Definition 2.5) to annotated versions of them (Definition 3.6). We start with the notion of a temporal expression which may be used to denote time points.

Definition 3.1 (Temporal expression) *(1) Every integer is a temporal expression. (2)* $\mathtt{X}_{now}$ *is a temporal expression. (3) If te_1, te_2 are temporal expressions, then $(te_1 + te_2)$ is a temporal expression.*

For example, 5, $\mathtt{X}_{\mathsf{now}} + 3$ and $\mathtt{X}_{\mathsf{now}} + \mathtt{X}_{\mathsf{now}} + 13$ are all temporal expressions. We will assume that there is an oracle that automatically assigns a value to $\mathtt{X}_{\mathsf{now}}$ (in an implementation, this could be done by looking at the system clock). Hence, a temporal expression can always be evaluated to a value. We now define temporal constraints.

Definition 3.2 (Temporal constraint (tc), Sol(tc)) *If te_1, te_2 are temporal expressions, then $te_1 \leq \mathsf{t} \leq te_2$ is an* atomic temporal constraint, *denoted* tc, *with free variable* t.

(1) Every atomic temporal constraint is a temporal constraint. (2) If $\mathsf{tc}_1, \mathsf{tc}_2$ are temporal constraints with the same free variable t, *then $\mathsf{tc}_1 \wedge \mathsf{tc}_2$ is a temporal constraint (with free variable* t*).*

For each temporal constraint tc *we denote by* Sol(tc) *the set of all solutions of* tc *(timepoints).*

For example, $\mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7$ is a temporal constraint. However, $\mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t}_1 \leq \mathtt{X}_{\mathsf{now}} + 7 \wedge \mathtt{X}_{\mathsf{now}} \leq \mathsf{t}_2 \leq \mathtt{X}_{\mathsf{now}} + 2$ is not a temporal constraint because it contains two distinct temporal variables. Intuitively, temporal constraints specify (implicitly) a set of *integer* time points, viz. the *integer* solutions of the temporal constraints. For space reasons, we will not formally define solutions of a temporal constraint, but appeal to the reader's intuition, and note that we are only interested in integer solutions of temporal constraints.

[1]Note that due to our simplified example, only the **Do** modality will be used in the sequel. We also focus on explaining the temporal-probabilistic aspects, and not the deontic subtleties associated with the modalities (these are explained in detail in [21]).

Thus, if $\mathtt{X}_{\mathsf{now}} = 3$, then the solutions of $\mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7$ are $\{5, 6, 7, 8, 9, 10\}$.

We assume the existence of a special set of variables called *probabilistic variables* and denoted by $\mathbf{X_i}$, that range over *real values* in the unit interval $[0, 1]$. We also assume the existence of some set of function symbols, each with an associated arity—these function symbols are pre-interpreted and map $[0, 1]^a$ to $[0, 1]$ for appropriate arities a.

Definition 3.3 (Probabilistic item ℓ) *(1) Every real number in the unit interval $[0, 1]$ is a probabilistic item. (2) Every probabilistic variable $\mathbf{X_i}$ is a probabilistic item. (3) If $\ell_1, \ldots, \ell_n$ are probabilistic items and f is a n-ary probabilistic function symbol, then $f(\ell_1, \ldots, \ell_n)$ is a probabilistic item.*

For example, if $\mathbf{X}$ is a probabilistic variable, then $\frac{\mathbf{X}+1}{2}$ is an example of a probabilistic item.

Definition 3.4 (Probability distribution function) *Suppose S is a nonempty set of random variables. A probability distribution function (**pdf**) w.r.t. S is a mapping δ from S to $[0, 1]$ such that $\Sigma_{s \in S}\delta(s) = 1$.*

We are now putting all these ingredients together. They constitute the temporal probabilistic information that is not available in the original basic framework of code calls.

Definition 3.5 (Annotation) *An annotation is a 5-tuple $[\otimes, \mathsf{tc}, \ell, \ell', \delta]$ where $\otimes$ is a probabilistic conjunction strategy,* tc *is a temporal constraint, ℓ, ℓ' are probabilistic items, and δ is a pdf over* Sol(tc)*. It is ground if ℓ, ℓ' are ground.*

For example, $[\oplus_{ig}, \mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7, \mathbf{X}, \frac{\mathbf{X}+1}{2}, \delta_u]$ is an annotation where $\oplus_{ig}$ represents the "ignorance" conjunction strategy and δ_u represents the uniform distribution. This annotation is not ground due to the presence of the variable $\mathbf{X}$. However, $[\oplus_{ig}, \mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7, 0.3, 0.5, \delta_u]$ is ground (despite the occurrence of t in it).

We are now ready to define a temporal probabilistic code call condition. We also define an extension of *status conjunctions* (which are just conjunctions of status atoms) to the temporal, probabilistic case.

Definition 3.6 (Annotated –ccc, ASC) *If χ is a (ground) code call condition, $(A_1 \wedge \ldots \wedge A_n)$ is a (ground) status conjunction, and $[\otimes, \mathsf{tc}, \ell, u, \delta]$ is a (ground) annotation, then $\chi : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ is a (ground)* annotated code call condition, *and $(A_1 \wedge \ldots \wedge A_n) : [\otimes, \mathsf{tc}, \ell, u, \delta]$ is a (ground)* annotated status conjunction *(*ASC*).*

$\mathbf{in}(\mathtt{c}, \mathtt{d} : f(\mathtt{a}, \mathtt{b})) : [\oplus_{ig}, \mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7, \mathbf{X}, \frac{\mathbf{X}+1}{2}, \delta_u]$ is an annotated code call condition. It says that there is a probability in the interval $[\mathbf{X}, \frac{\mathbf{X}+1}{2}]$ that at some time point between $\mathtt{X}_{\mathsf{now}} + 2$ and $\mathtt{X}_{\mathsf{now}} + 7$, the code call $\mathtt{d} : f(\mathtt{a}, \mathtt{b})$ will return c in its output. Furthermore, for any specific time point in this time interval, the specific probability that c will be returned is uniformly distributed. Similarly,

$$\big(\mathbf{Do}\ buy(\mathtt{ibm}) \wedge \mathbf{Do}\ sell(\mathtt{lucent})\big) : [\oplus_{ig}, \mathtt{X}_{\mathsf{now}} + 2 \leq \mathsf{t} \leq \mathtt{X}_{\mathsf{now}} + 7, \mathbf{X}, \tfrac{\mathbf{X}+1}{2}, \ell]$$

is an annotated status conjunction. It says that there is a probability in the interval $[\mathbf{X}, \frac{\mathbf{X}+1}{2}]$ that at some time point between $\mathtt{X}_{\mathsf{now}} + 2$ and $\mathtt{X}_{\mathsf{now}} + 7$, the agent will both buy IBM stock and sell Lucent stock. Furthermore, the probability

of both events occurring is computed from their individual probabilities by using the conjunction strategy of ignorance.

Annotated ccc's and annotated status conjunctions are the main ingredients to formalize conditions as to which actions should be executed given the current state of the world. The rules of a program determine under which conditions a particular status is given to an action. The overall semantics of TP-agents (to be defined in the next section) determines which set of actions should be executed in the current state.

Definition 3.7 (TP-rule) *A (ground)* temporal probabilistic *(*TP*) rule is an expression of the form* $SA_0 \leftarrow accc_1 \wedge \ldots \wedge accc_n \wedge asc_1 \wedge \ldots \wedge asc_n$, *where* SA_0 *is a (ground) annotated status conjunction containing only one status atom in it,* $accc_1, \ldots, accc_m$ *are (ground) annotated code call conditions and* $asc_1, \ldots, asc_n$ *are (ground) annotated status conjunctions (*ASC*).*

A temporal probabilistic *(*TP*) rule is positive if any annotation that appears in the rule and is associated with a status atom is of the form* $[\otimes, \mathsf{tc}, 1, 1, \delta]$.

Note that the restriction to the intervall [1,1] above is only imposed on status atoms, and not on annotated ccc's. This is to make sure that there is no uncertainty in the actions to be computed but only in the state.

Definition 3.8 (TP Agent Program (TPP)) *A* temporal probabilistic agent program *(*TPP *for short) is a finite set of* TP *rules. It is positive if all its rules are positive.*

Example 3.9 (Stock example revisited) *The following two rules encode the simple stock example described earlier.*

$$\begin{array}{l}\textbf{Do}\ buy(\mathtt{X}) : [\oplus_{ig}, \mathtt{X}_{now} \le \mathsf{t} \le \mathtt{X}_{now} + 5, 1, 1, \delta_u] \leftarrow \\ \quad \textbf{in}(\mathtt{X}, \mathsf{stock}: myportfolio()) \quad : [\oplus_{ig}, \mathtt{X}_{now}, 1, 1, \delta_u] \ \wedge \\ \quad \textbf{in}(\text{"}yes\text{"}, \mathsf{stock}: over(\mathtt{X}, 50)) \quad : [\oplus_{ig}, \mathtt{X}_{now} + 10 \le \mathsf{t} \\ \qquad\qquad \le \mathtt{X}_{now} + 20, 0.8, 1, \delta_u] \\ \textbf{Do}\ buy(\mathtt{X}) : [\oplus_{ig}, \mathtt{X}_{now} \le \mathsf{t} \le \mathtt{X}_{now} + 3, 1, 1, \ell] \leftarrow \\ \quad (\textbf{in}(\text{"}no\text{"}, \mathsf{stock}: myportfolio()) \ \wedge \\ \quad\ \textbf{in}(\text{"}ok\text{"}, \mathsf{stock}: diversify(\mathtt{X}))) : [\oplus_{ig}, \mathtt{X}_{now}, 1, 1, \ell] \ \wedge \\ \quad\ \textbf{in}(\text{"}yes\text{"}, \mathsf{stock}: over(\mathtt{X}, 50)) : [\oplus_{ig}, \mathtt{X}_{now} + 10 \le \mathsf{t} \\ \qquad\qquad \le \mathtt{X}_{now} + 20, 0.9, 1, \delta_u]\end{array}$$

The first rule says that if stock X is in my current portfolio and it is expected (with 80% probability or more) to go over $ 50 per share sometime between day 10 and 20 from now, then buy this stock in the next 5 days. The second rule says that if the stock is not in my current portfolio and acquiring it is consistent with the goal to maintain a diversified portfolio, then we should buy the stock in the next 3 days as long as there is an over 90% probability of its going up to $ 50 per share.

4. SEMANTICS OF TP AGENTS

Though the reader may be tempted to infer that the rules given above are read in terms of usual logical inference, it will soon be clear that things are somewhat more complex.

4.1 Satisfaction of annotated ccc's

Definition 4.1 (Possible answer situations) *Consider an agent state* $\mathcal{O}$, *a code call cc, a set* T *of time points, and an object o whose type is the same as cc's output type. The* possible answer situations *of cc w.r.t.* T *and o, denoted* $\mathsf{pas}(cc, o, \mathsf{T})$ *is the set* $\{(\mathsf{t}, Obj, \delta) \mid \mathsf{t} \in \mathsf{T}$ *and* $(Obj, \delta) \in cc^{\mathrm{TP}}(\mathsf{t})$ *and* $o \in Obj\}$.

It is easy to see that when T is a singleton, because of the coherence requirement on TPCC's, $\mathsf{pas}(cc, o, \mathsf{T})$ is either the empty set or a singleton.

Definition 4.2 (Answer time probabilities) *Consider an agent state* $\mathcal{O}$, *a code call cc, a time point* t, *and an object o whose type is the same as cc's output type. The probability that o is in the answer of cc at time* t, *denoted* $\mathsf{prob}(o, cc, \mathsf{t})$, *is given by:*

$$\mathsf{prob}(o, cc, \mathsf{t}) = \begin{cases} \delta(o) & \text{if } \mathsf{pas}(cc, o, \{\mathsf{t}\}) = \{(\mathsf{t}, Obj, \delta)\} \\ 0 & \text{otherwise.} \end{cases}$$

The probability with which a state $\mathcal{O}$ satisfies a code call condition χ at time t under conjunction strategy $\otimes$, denoted $\mathsf{prob}(\chi, \mathcal{O}, \mathsf{t})$ is given by $\otimes_{\mathbf{in}(o,\, cc) \in \chi} \mathsf{prob}(o, cc, \mathsf{t})$, where $\otimes$ is the conjunction strategy associated with χ.

However, when we look at a ground annotated code call of the form $\mathbf{in}(o, cc) : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$, we may need to consider multiple time points. For example, tc may have two solutions, $\mathsf{t} = t_1$ and $\mathsf{t} = t_2$. The probability that o is in the answer returned by cc is the probability that o is in the answer returned by cc at time t_1 (event e_1) **or** at time t_2 (event e_2). However, we have no information about dependencies between these two events. Are they independent? Is there some kind of correlation between these events? Are we completely ignorant about the relationship between these two events? In general, we are attempting to evaluate the probability of a disjunction of two events, given information about the probability of the individual events involved. To do this, we assume that every agent has an arbitrary but fixed probabilistic disjunction strategy $\oplus$ that it uses.

Definition 4.3 (Satisfaction of ASC's) *Suppose* $\mathcal{O}$ *is an agent state and* $\oplus$ *is a fixed probabilistic disjunction strategy.*

1. $\mathcal{O}$ *satisfies a ground annotated code call condition* $\chi : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$, *denoted* $\mathcal{O} \models \chi : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ *iff* $\ell \le \oplus\{\mathsf{prob}(\chi, \mathcal{O}, \mathsf{t}) \mid \mathsf{t} \in Sol(\mathsf{tc})\} \le \ell$,
2. $\mathcal{O}$ *satisfies a ground annotated code call condition* $\chi : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ *with respect to* t_{now} *and the past, denoted* $\mathcal{O} \models^{\mathsf{now}} \chi : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ *iff* $\ell \le \oplus\{\mathsf{prob}(\chi, \mathcal{O}, \mathsf{t}) \mid \mathsf{t} \in Sol(\mathsf{tc}), \mathsf{t} \le \mathsf{t}_{now}\} \le \ell$,
3. $\mathcal{O}$ *satisfies* $accc_1 \wedge accc_2$ *iff* $\mathcal{O} \models accc_1$ *and* $\mathcal{O} \models accc_2$.
4. $\mathcal{O}$ *satisfies* $(\forall x)F$ *iff* $\mathcal{O} \models F[x/c]$ *where c is a constant of the same type as x and* $F[x/c]$ *denotes the replacement of all free*[2] *occurrences of x in F by c.*

The definition of the last two cases for $\models^{\mathsf{now}}$ *is similar to that of* $\models$.

4.2 TP status models

We now define the formal semantics of TP-agents.

Definition 4.4 (TP-status interpretation (TPSI)) *A* TP-status interpretation *(*TPSI *for short) is a mapping* ρ *that maps natural numbers to status sets.*

[2]Due to space restrictions, we do not formally define free occurrences here but refer the reader to the standard definition which may be readily adapted to our case.

Thus, given any time point t, $\rho(\mathsf{t})$ is a status set. We now define what it means for a TP status interpretation to satisfy an annotated status conjunction.

Definition 4.5 (Satisfaction of annotated status conj.) *Suppose ρ is a* TP*-status interpretation. The probability with which ρ satisfies a status atom $\mathsf{Op}\alpha$ at time t is*

$$\mathsf{prob}(\mathsf{Op}\alpha, \rho, \mathsf{t}) = \begin{cases} 1 & \text{if } \mathsf{Op}\alpha \in \rho(\mathsf{t}) \\ 0 & \text{otherwise.} \end{cases}$$

The probability with which ρ satisfies a status conjunction $\varrho = (A_1, \wedge \ldots A_n)$ w.r.t. a given conjunction strategy $\otimes$ and at a given time t, denoted $\mathsf{prob}_{\oplus}(\varrho, \rho, \mathsf{t})$ is given by:

$$\mathsf{prob}(A_1, \rho, \mathsf{t}) \otimes \mathsf{prob}(A_2, \rho, \mathsf{t}) \otimes \cdots \otimes \mathsf{prob}(A_n, \rho, \mathsf{t}).$$

Let $(A_1 \wedge \ldots \wedge A_n) : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ be a status conjunction. ρ satisfies $(A_1 \wedge \ldots \wedge A_n) : [\otimes, \mathsf{tc}, \ell, \ell', \delta]$ at time t w.r.t. $\oplus$ iff $\ell \leq \oplus\{\mathsf{prob}_{\oplus}(A_1 \wedge \ldots \wedge A_n, \rho, \mathsf{t}) \mid \mathsf{t} \in \mathit{Sol}(\mathsf{tc})\} \leq \ell'$.

Note that for status *atoms*, a TPSI either satisfies it (probability 1) or not (probability 0). For status conjunctions however, other probabilities can arise as well. Due to space restrictions, we do not consider the case when a TPSI assigns other probabilities to status atoms.

Definition 4.6 *Suppose $SA_0 \leftarrow accc_1 \wedge \ldots \wedge accc_n \wedge asc_1 \wedge \ldots \wedge asc_n$ is a* ground TP*-rule, ρ is a* TP*-status interpretation and $\mathcal{O}$ is the current agent state.*

ρ satisfies *the above ground rule in state $\mathcal{O}$ iff either:*

1. *$\mathcal{O}$ does not satisfy $accc_1 \wedge \ldots \wedge accc_n$ with respect to now and the past, or*
2. *ρ does not satisfy $asc_1 \wedge \ldots \wedge asc_n$ or*
3. *ρ satisfies SA_0.*

ρ satisfies $\mathcal{TP}$, where $\mathcal{TP}$ is a TPP*, iff for each temporal probabilistic agent rule $r \in \mathcal{TP}$: ρ satisfies r.*

An agent may record the actions it took (or was obliged to take, forbidden from taking etc.) in the past. This leads to the notion of an *action history*.

Definition 4.7 (Action History acthist) *An* action history *acthist for an agent is a partial function from $\mathbb{N}$ to status sets satisfying $\mathit{acthist}(\mathsf{t}) = \emptyset$ for all $\mathsf{t} > \mathsf{t}_0$ for a $\mathsf{t}_0 \in \mathbb{N}$.*

Intuitively, an action history specifies not only what the agent has done in the past, but also what an agent is *obliged/-permitted* to or *forbidden* from doing in the future. As action histories are partial (rather than total) functions, the agent developer has the flexibility to choose to store none, some or all status atoms associated with the agent's past.

An action history and a TP-status interpretation both make statements about action status atoms. Therefore they need to be *compatible*.

Definition 4.8 (History-Compatible TPSI) *Suppose the current time is t_{now} and $\mathit{acthist}(\cdot)$ denotes the action history of an agent, and suppose ρ is a* TPSI*. ρ is said to be* action history-compatible *at time t_{now} iff for all $\mathsf{t} < \mathsf{t}_{now}$, if $\mathit{acthist}(\mathsf{t})$ is defined, then $\rho(\mathsf{t}) = \mathit{acthist}(\mathsf{t})$, and for all $\mathsf{t} \geq \mathsf{t}_{now}$, if $\mathit{acthist}(\mathsf{t})$ is defined, then $\mathit{acthist}(\mathsf{t}) \subseteq \rho(\mathsf{t})$.*

In other words, for a TPSI to be compatible with an action history, it must be consistent with the past history of actions taken by the agent and with commitments to do things in the future that were made in the past by the agent.

4.3 Feasible Temporal Probabilistic Interpr.

Let us consider an agent $\mathfrak{a}$ that uses a TP-interpretation ρ to determine what actions it should take, and when it should take these actions. Not all such interpretations make sense. For example we must impose conditions that ensure an action is not both permitted and forbidden at the same time (deontic conditions as formalized in Definitions 4.11, 4.10). And such an interpretation ρ must be closed under the rules of the TP program. The main results of this section are Definition 4.13, which states the conditions we need, and Theorem 4.14, which determines the complexity of verifying them for a given TP-interpretation.

We also have to make sure that actions scheduled for the future can be executed. To this end, we have to have a model about the expected states in the future.

Definition 4.9 (Expected States at time t: $\mathcal{EO}(\mathsf{t})$) *Suppose the current time is t_{now}, $\mathcal{O}$ is the agent state and ρ is a* TP*-status interpretation. The agent's expected states are defined as follows: $\mathcal{EO}(\mathsf{t}_{now}) = \mathcal{O}$,*

- *For all time points $\mathsf{t} > \mathsf{t}_{now}$, $\mathcal{EO}(\mathsf{t})$ is the result of concurrently executing $\{\alpha \mid \mathbf{Do}\,\alpha \in \rho(\mathsf{t}-1)\}$ in state $\mathcal{EO}(\mathsf{t}-1)$.*

The expected states are used in the next definition (last condition) to ensure actions can be executed.

Given a set S of status atoms, let $\mathbf{D\text{-}Cl}(S)$ be the smallest superset S' of S such that $\mathbf{O}\alpha \in S' \rightarrow \mathbf{P}\alpha \in S'$. Likewise, let $\mathbf{A\text{-}Cl}(S)$ be the smallest superset S^* of S such that (i) $\mathbf{O}\alpha \in S^* \rightarrow \mathbf{Do}\,\alpha \in S^*$ and (ii) $\mathbf{Do}\,\alpha \in S^* \rightarrow \mathbf{P}\alpha \in S^*$. We say that set S is *deontically closed iff $S = \mathbf{D\text{-}Cl}(S)$* and *action closed iff $S = \mathbf{A\text{-}Cl}(S)$*.

Definition 4.10 (TP Deontic Consistency) *Suppose $\mathcal{O}$ is the agent state. A* TP*-status interpretation ρ is said to be* TP deontically consistent *at time t_{now} iff it satisfies the following conditions for all time points t (in the following, $Pre(\alpha)$ stands for the preconditions of the action α):*

(1) $\mathbf{O}\alpha \in \rho(\mathsf{t}) \rightarrow \mathbf{W}\alpha \notin \rho(\mathsf{t})$;
(2) $\mathbf{P}\alpha \in \rho(\mathsf{t}) \rightarrow \mathbf{F}\alpha \notin \rho(\mathsf{t})$;

if $\mathsf{t} \leq \mathsf{t}_{now}$, $\mathbf{P}\alpha \in \rho(\mathsf{t})$, then $\mathsf{prob}(Pre(\alpha), \mathcal{O}, \mathsf{t}) = 1$,

if $\mathsf{t} > \mathsf{t}_{now}$, $\mathbf{P}\alpha \in \rho(\mathsf{t})$, then $\mathsf{prob}(Pre(\alpha), \mathcal{EO}(\mathsf{t}), \mathsf{t}) = 1$.

Thus, if $\rho(4) = \{\mathbf{Do}\,\alpha, \mathbf{F}\alpha\}$, then ρ cannot be TP-deontically consistent. The following definition explains what it means for a TP-status interpretation to be closed under the deontic modalities and under actions.

Definition 4.11 (TP Deontic/Action Closure) *ρ is said to be* TP deontically closed *at time t_{now} iff for all time points t: $\mathbf{D\text{-}Cl}(\rho(\mathsf{t})) = \rho(\mathsf{t})$. ρ is said to be* TP action closed *at time t_{now} iff for all time points t: $\mathbf{A\text{-}Cl}(\rho(\mathsf{t})) = \rho(\mathsf{t})$.*

For a temporal status set to be feasible, it must satisfy the additional requirement of TP-state consistency. We assume

that an agent has a set of integrity constraints $\mathcal{IC}$ of the form: $\psi \Rightarrow \chi$ where ψ is a code call condition, and χ is an atomic code call. These constraints are evaluated in the state. They say that an agent must never transition to a state that violates any of these constraints. [3]

Definition 4.12 (TP State Consistency) *ρ is said to be* TP-state consistent at time $\mathsf{t_{now}}$ *iff for each integrity constraint $\psi \Rightarrow \chi$ in $\mathcal{IC}$ for all $\mathsf{t} \leq \mathsf{t}_{now}$ for every legal assignment of objects from $\mathcal{O}$ to the variables of ψ and χ either* $\mathsf{prob}(\psi, \mathcal{O}, i) \neq 1$ *or* $\mathsf{prob}(\chi, \mathcal{O}, \mathsf{t}) = 1$

A feasible TPSI is like a model of a classical logic theory.

Definition 4.13 (Feasible TPSI (FTPSI)) *Suppose the current time is t_{now}, $\mathcal{TP}$ is a* TPP, *$\mathcal{O}$ is an agent state, acthist is an action history and $\mathcal{IC}$ is a set of integrity constraints. A* TP-*status interpretation ρ satisfying $\rho(i) \neq \emptyset$ for only finitely many i is said to be* feasible *with respect to the above parameters, denoted by* FTPSI, *iff*

(1) ρ satisfies all rules in $\mathcal{TP}$,

(2) ρ is TP *deontically consistent at time t_{now},*

(3) ρ is TP *deontically and action closed at time t_{now},*

(4) ρ is TP*-state consistent at time t_{now},*

(5) ρ is history compatible at time t_{now}.

The idea behind FTPSI's is that whenever a state change occurs (e.g. when the agent receives a message, or when the clock ticks, or when an update is made), the agent computes a feasible TPSI ρ and concurrently executes all actions α such that $\mathbf{Do}\,\alpha \in \rho(\mathsf{t_{now}})$. An agent may have zero, one, or many FTPSI's in a given state.

Theorem 4.14 (Complexity) *Suppose the current time is t_{now}, $\mathcal{TP}$ is a* TPP, *$\mathcal{O}$ is an agent state, acthist is an action history and $\mathcal{IC}$ is a set of integrity constraints. The problem of checking whether a given* TPSI *ρ is feasible or not, is **NP**-complete.*

5. TPSI COMPUTATION

The preceding section defines a formal semantics based on the concept of a feasible TP-status interpretation. We now show how FTPSI's can be constructed for finite positive TPP's.

Since we are considering only positive rules, for simplicity, we assume that an annotation associated with a status conjunction is of the form [tc] where tc is a temporal constraint. In addition, if Sol(tc) is a singleton $\{t\}$, we will use the annotation $[t]$. We will refer to status atoms of the form $\mathsf{Op}(\alpha)[\mathsf{tc}]$ as *simple annotated status atoms* and to status atoms of the form $\mathsf{Op}(\alpha)[\mathsf{t}]$, where t is an integer, as *singleton annotated status atoms*. It is straightforward to construct a TP-status interpretation from a set of singleton annotated status atoms sa-$\mathcal{TS}$: $\mathsf{Op}(\alpha) \in \rho(\mathsf{t})$ *iff* $\mathsf{Op}(\alpha)[\mathsf{t}] \in$ sa-$\mathcal{TS}$. In this case, ρ and sa-$\mathcal{TS}$ are *compatible*.

[3] [10] also allows agents to have a set, possibly empty, of *action constraints*. It has been shown [21] that action constraints can be expressed as integrity constraints, and hence, we do not consider them in this paper.

Furthermore, we will say that sa-$\mathcal{TS}$ has a given property, e.g., it is feasible, *iff* its compatible ρ is feasible. Thus, given a TPP, $\mathcal{TP}$, an agent state, $\mathcal{O}$, and an action history acthist, our goal is to construct a feasible sa-$\mathcal{TS}$. We will use sa-$\mathcal{TS}$ and its corresponding ρ exchangely.

However, if sa-$\mathcal{TS}$ is a general simple annotated status atoms set, there are several TP-status interpretations that can be constructed based on it and the notion of compatibility is defined as follows.

Definition 5.1 (TPSI Compatible with sa-$\mathcal{TS}$) *A* TP-*status interpretation ρ is* compatible with *sa-$\mathcal{TS}$ iff for every $\mathsf{Op}\alpha[\mathsf{tc}]$ in sa-$\mathcal{TS}$, there is a solution $\mathsf{t} = i$ of tc such that $\mathsf{Op}\,\alpha \in \rho(i)$.*

There are an infinite number of TP-status interpretations that are compatible with this sa-$\mathcal{TS}$. On the other hand, there are infinitely many general simple annotated status atoms sets that are compatible with a given ρ. We denote the subset of the singleton status atoms of sa-$\mathcal{TS}$ by $Singl(\text{sa-}\mathcal{TS})$.

5.1 The Fixed point operator

The main difficulty is to ensure that the FTPSI to be constructed is closed under the rules of $\mathcal{TP}$. This is because when a rule causes atoms to be added to $\rho(i)$, for some $i \geq \mathsf{t_{now}}$, it may cause other rules to fire, which may cause other atoms to be added to $\rho(i)$, for some $i \geq \mathsf{t_{now}}$. This in turn may cause additional rules to fire.

This difficulty suggests taking advantage of well-known methods from logic programming [20], namely to construct a suitable monotone fixpoint operator and to relate its least fixpoint $\mathsf{D}_{\mathcal{TP}}\uparrow^{\omega}$ with feasible TP-status interpretations. Thus, we first define a fixpoint operator. The definition uses the notion of modalities implying modalities which is defined as: **O** implies both **Do** and **P**, **Do** implies **P** and all modalities imply themselves.

Definition 5.2 (Operator $\mathbf{D}_{\mathcal{TP}}$) *Let $\mathcal{TP}$ be a* TPP, *$\mathcal{O}$ a state and sa-$\mathcal{TS}$ a set of simple annotated status atoms. Then we define $D_{\mathcal{TP}}$(sa-$\mathcal{TS}$) to be the set*

$\{\mathsf{Op}'\alpha[\mathsf{tc}'] \mid$ $\mathsf{Op}\alpha : [\mathsf{tc}] \leftarrow \mathsf{accc}_1 \wedge \ldots \wedge \mathsf{accc}_n \wedge \varrho_1 : [\mathsf{tc}_1] \wedge \ldots \wedge \varrho_m : [\mathsf{tc}_m]$
is a ground instance of a rule in $\mathcal{TP}$ and
(I) for all $1 \leq i \leq n$*: $\mathcal{O} \models^{\mathsf{now}} \mathsf{accc}_i$* ***and***
(II) for all $1 \leq i \leq m$*:*
If $\varrho_i = \mathsf{Op}_i\alpha_i$
then *there exists $\mathsf{Op}'_i\alpha_{i_1}[\mathsf{tc}'_i]$ in sa-$\mathcal{TS}$ s. t.:*
(1) Op'_i implies Op_i,
(2) $\mathsf{Sol}(\mathsf{tc}'_i) \subseteq \mathsf{Sol}(\mathsf{tc}_i)$ ***and***
If $\varrho_i = \mathsf{Op}_{i_1}\alpha_{i_1} \wedge \ldots \wedge \mathsf{Op}_{i_k}\alpha_{i_k}$
then *there exist $\mathsf{t}_i \in \mathsf{Sol}(\mathsf{tc}_i)$ and for $1 \leq j \leq k$: $\mathsf{Op}'_{i_j}\alpha_{i_j}[\mathsf{t}_i]$ in sa-$\mathcal{TS}$ s. t. Op'_{i_j} implies Op_{i_j}.* ***and***
(III) *$\mathsf{tic} = [\mathsf{tc} \wedge t \geq \mathsf{t_{now}}]$ and Op implies Op'.* $\}$

In order to find a fixed point we need to iterate the operator. However, we do not start the operator at $\emptyset$: this is because part of the TP-status interpretation we want to construct is already determined by acthist. Therefore we define sa-$\mathcal{TS}_{\text{start}}$ to include the set of singleton annotated status atoms that corresponds to acthist. In some situations

we will add to sa-$\mathcal{TS}_{\text{start}}$ status atoms that may belong to the feasible TP-status interpretation.

Definition 5.3 (Iterations of $\mathsf{D}_{\mathcal{TP}}$) *Let $\mathcal{TP}$ be a positive* TPP*, and $\mathcal{O}$ be an agent state. The* iterations *of $D_{\mathcal{TP}}$ are defined as follows:*

$$\begin{aligned} D_{\mathcal{TP}} \uparrow^0 &= \textit{sa-}\mathcal{TS}_{start}. \\ D_{\mathcal{TP}} \uparrow^{(j+1)} &= D_{\mathcal{TP}}(D_{\mathcal{TP}} \uparrow^j). \\ D_{\mathcal{TP}} \uparrow^\omega &= \bigcup_j D_{\mathcal{TP}} \uparrow^j . \end{aligned}$$

When sa-$\mathcal{TS}_{\text{start}}$ is not clear from the context, we will use the notation $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$ (sa-$\mathcal{TS}_{\text{start}}$) to explicitly specify it.

Theorem 5.4 *$D_{\mathcal{TP}}$ is a monotone and continuous operator. Hence, $D_{\mathcal{TP}} \uparrow^\omega$ is its least fixpoint.*

We are now ready to show that $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$ has the properties of TP deontic and action closure, and also that all feasible temporal status sets must be compatible with $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$. These properties will later help us in computing feasible temporal status sets.

Theorem 5.5 *Let $\mathcal{TP}$ be a positive* TPP*, $\mathcal{O}$ a state.*

1. *There is a* TPSI *ρ compatible with $D_{\mathcal{TP}} \uparrow^\omega$ which is* TP *deontically closed and temporally action closed.*
2. *If ρ is a feasible* TPSI*, then it is compatible with $D_{\mathcal{TP}} \uparrow^\omega$.*

Given the above theorem, it seems that a TPSI that is compatible with $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$ is a good candidate for constructing a feasible TP-status interpretation. This is particularly true if $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$ is a set of singleton annotated status atoms. However, $\mathsf{D}_{\mathcal{TP}} \uparrow^\omega$ may be only a set of simple annotated status atoms and we will need to choose one of the TPSI's that are compatible with it. For a precise description of how to make such a choice, let us introduce the concept of a TP hitting set.

Definition 5.6 (TP Hitting Set) *Suppose sa-$\mathcal{TS}$ is a set of simple annotated status atoms. A* TP hitting set*, H, for sa-$\mathcal{TS}$ is a minimal set of singleton ground annotated status atoms of the form $\mathsf{Op}\alpha[i]$ such that:*

> *For every simple annotated status atom $\mathsf{Op}\alpha[\mathsf{tc}] \in$ sa-$\mathcal{TS}$, there is an annotated atom of the form $\mathsf{Op}\alpha[i]$ in H such that i is a solution of tc, and if $i < \mathsf{t_{now}}$, then $\mathsf{Op}\alpha \in \mathsf{acthist}(i)$.*

We use chs(sa-$\mathcal{TS}$) to denote the set of all TP *hitting sets for sa-$\mathcal{TS}$.*

We will use a subroutine called *find_member_chs*(sa-$\mathcal{TS}$) which finds a member of chs(sa-$\mathcal{TS}$) that is not a subset of sa-$\mathcal{TS}$. If no such element exists, it returns "No." We do not specify the implementation of this algorithm as it can be easily implemented (using standard hitting set algorithms [3]). Before presenting our algorithm, we need an additional definition.

Definition 5.7 (Solution Closed) *A set F of simple annotated status atoms is said to be* solution-closed *iff*

for all simple annotated status atoms $\mathsf{Op}\alpha[\mathsf{tc}] \in F$, the following holds: tc has a solution i and $\mathsf{Op}\alpha[i] \in F$

5.2 Feasible Temporal Status Set Algorithm

Given a finite set of singleton annotated status atoms sa-$\mathcal{TS}$ that is closed under the program rules of $\mathcal{TP}$, it is possible to check whether it is feasible. We will assume that there is a **FeasTPI** algorithm that checks

1. for all $i \leq \mathsf{t_{now}}$, for all status atoms $\mathsf{Op}\alpha[i]$ in sa-$\mathcal{TS}$ where $\mathsf{Op} \in \{\mathbf{P}, \mathbf{O}, \mathbf{Do}\}$: $\mathsf{prob}(Pre(\alpha), \mathcal{O}, i) = 1$,
2. for all $i > \mathsf{t_{now}}$, for all status atoms $\mathsf{Op}\alpha[i]$ in sa-$\mathcal{TS}$ where $\mathsf{Op} \in \{\mathbf{P}, \mathbf{O}, \mathbf{Do}\}$: $\mathsf{prob}(Pre(\alpha), \mathcal{EO}(i), i) = 1$,
3. temporal deontic consistency for sa-$\mathcal{TS}$.

It returns **true** if all these requirements are met—otherwise it returns *false*. It is possible to develop such algorithm since we require TPSI's to be finite. Thus, we only need to check these conditions for finitely many i.

Our main procedure for computing FTPSI's is recursive. It takes three arguments: a set of simple annotated atoms sa-$\mathcal{TS}_{\text{start}}$, a TPP $\mathcal{TP}$ and the agent state, $\mathcal{O}$. In the first invocation of this procedures, sa-$\mathcal{TS}_{\text{start}}$ is the set corresponding to acthist, i.e.,

$$\textsf{sa-}\mathcal{TS}_{\text{start}} := \bigcup_{\{i \text{ s.t. } \mathsf{acthist}(i) \text{ is defined}\}} \{\mathsf{Op}\alpha[i] \mid \mathsf{Op}\alpha \in \mathsf{acthist}(i)\}.$$

As the procedure is recursively invoked, new simple annotated status atoms may be added to sa-$\mathcal{TS}_{\text{start}}$.

Algorithm 5.8 ***ComputeTPI(*sa-$\mathcal{TS}_{start}$*,$\mathcal{TP}$, $\mathcal{O}$)***

1. *sa-$\mathcal{TS}_{new}$ = $D_{\mathcal{TP}} \uparrow^\omega$ (sa-$\mathcal{TS}_{start}$).*
2. ***If** sa-$\mathcal{TS}_{new}$ is solution closed **then***
   (a) ***if FeasTPI**($Singl$(sa-$\mathcal{TS}_{new}$), $\mathcal{O}$) **then** return $Singl$(sa-$\mathcal{TS}_{new}$).*
   (b) ***else** return false.*
3. ***Else** HS = find_member_chs(sa-$\mathcal{TS}$).*
4. ***While** $HS \neq \emptyset$ **do***
   (a) $H = head(HS), HS = HS \setminus \{H\}$.
   (b) *pos=**ComputeTPI(** sa-$\mathcal{TS}_{new} \cup H, \mathcal{TP}, \mathcal{O}$).*
   (c) ***If** pos $\neq$ false **then** return pos.*
5. *Return false.*

The intuition behind the algorithm is as follows. It starts with acthist and closes it under program rules using the $\mathsf{D}_{\mathcal{TP}}$ operator. If the result is solution closed, then we check for feasibility. If not, there may be many "reasons" why the result is not solution closed. For example, $\mathbf{Do}\,\alpha_1[\mathsf{tc}_1]$ and $\mathbf{Do}\,\alpha_2[\mathsf{tc}_2]$ may both be such that no status atom of the form $\mathbf{Do}\,\alpha_1[\mathsf{t}_1]$ and $\mathbf{Do}\,\alpha_2[\mathsf{t}_2]$ are present in sa-$\mathcal{TS}_{new}$ where $\mathsf{t}_1, \mathsf{t}_2$ are solutions of $\mathsf{tc}_1, \mathsf{tc}_2$ respectively. $\mathsf{tc}_1, \mathsf{tc}_2$ may have lots of solutions but we can pick a hitting set of their set of solutions and add those simple annotated status atoms to sa-$\mathcal{TS}_{new}$ to remove the "reasons" for solution closure to fail. This process may in turn may cause new status atoms to be derivable (i.e. sa-$\mathcal{TS}_{new}$ may not be closed under program rules after the addition of these atoms) and hence the process must be repeated till we reach a situation where termination occurs through step (2) of the algorithm. The following theorem states the correctness of the above procedure.

Theorem 5.9 (Soundness and Completeness) *The ComputeTPI (5.8) procedure generates a feasible* TP*-status interpretation (if one exists).*

6. RELATED WORK

There has not been a lot of work on agents that reason about actions, time and uncertainty. Haddawy [16] develops a logic for reasoning about actions, probabilities, and time. His framework is purely logical and does not deal with legacy code. In addition, he studies a general logic, whereas our TP agents have a restricted syntactic form which makes them more suitable for computation.

Lehmann and Shelah [19] were one of the first to integrate time and probability. They developed a probabilistic temporal logic - however, actions were not studied, and an algorithm by which an agent can decide what to do, given its operating principles, was not studied either. Dean and Kanazawa have also studied the integration of probability and time with a view to developing efficient planning techniques [4, 17]. Their main interest is in how probabilities of facts and events change over time. In our setting, where actions are also present, this would amount to metareasoning over histories and possible futures. In addition, Dean and Kanazawa attempt to model decreasing persistence of fluents – something we do not address here. Dubois and his colleagues [7] have studied the integration of uncertainty and time – they extend the well-known possibilistic logic theory [8] to a "timed possibilistic logic."

The work on *MetaTem* [1] and its successor, *Concurrent MetaTem* [13] are closely related to our work as they provide a framework to reason about agents with temporal reasoning capabilities. No probabilities are processed in their work—when probabilities are ignored, the relationship of their work to our framework has been cleanly described in [5].

Shoham and his colleagues [22] have developed a framework for integrating beliefs, time, commitment, desires and multiple agents. However, they do not deal with probabilities at all. Likewise, Sandewall [9] has presented a framework for integrating time and action, but does not deal with probabilities. Durfee and his colleagues [14] have developed a logic of knowledge and belief to model multiagent coordination. Their framework permits an agent to reason not only about the world and its own actions, but also to simulate and model the behavior of other agents in the environment. In a separate paper [15], they show how one agent can reason with a probabilistic view of the behavior of other agents so as to achieve coordination.

7. REFERENCES

[1] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: A framework for, programming in temporal logic. In *LNCS 430*. Springer-Verlag, June 1989.

[2] G. Boole. *The Laws of Thought*. Macmillan, London, 1854.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1989.

[4] T. Dean and K. Kanazawa. Probabilistic Temporal Reasoning. In *Proceedings AAAI*, pages 524–529, 1988.

[5] J. Dix, S. Kraus, and V. Subrahmanian. Temporal agent reasoning. *Artificial Intelligence*, 127(1):87–135, 2001.

[6] J. Dix, M. Nanni, and V. S. Subrahmanian. Probabilistic agent reasoning. *ACM Transactions of Computational Logic*, 1(2):201–245, 2000.

[7] D. Dubois, J. Lang, and H. Prade. Towards Possibilistic Logic Programming. In *Proceedings of the Eigth ICLP*, pages 581–595, Paris, France, June 1991. MIT Press.

[8] D. Dubois and H. Prade. Possibilistic logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3, Nonmonotonic and Uncertain Reasoning*, pages 439–513. Oxford University Press, 1994.

[9] E. Sandewall. Features and Fluents: The Representation of Knowledge about Dynamical Systems, 1994.

[10] T. Eiter, V. Subrahmanian, and G. Pick. Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence*, 108(1-2):179–255, 1999.

[11] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Massachusetts, 1995. 2nd printing.

[12] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, July/August 1990.

[13] M. Fisher. A survey of Concurrent METATEM —, the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the, First International Conference*. Springer-Verlag, July 1994.

[14] P. Gmytrasiewicz and E. Durfee. A Logic of Knowledge and Belief for Recursive Modeling. In *Proceedings of the 10th National Conference on Artificial, Intelligence*, pages 628–634, San Jose, CA, 1992. AAAI Press/MIT Press.

[15] P. Gmytrasiewicz, E. Durfee, and D. Wehe. A Decision-Theoretic Approach to Coordinating Multiagent, Interactions. In *Proceedings of the 12th IJCAI*, pages 62–68, Sydney, Australia, 1991. Morgan Kaufmann.

[16] P. Haddawy. *Representing Plans under Uncertainty: A Logic of Time, Chance and Action*. PhD thesis, University of Illinois, 1991.

[17] K. Kanazawa. A Logic and Time Nets for Probabilistic Inference. In *Proceedings AAAI-91*, pages 360–365, Anaheim, August 1991.

[18] V. S. Lakshmanan, N. Leone, R. Ross, and S. V. S. ProbView: A Flexible Probabilistic Database System. *ACM Transactions on Database Systems*, 22(3):419–469, September 1997.

[19] D. Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–198, 1982.

[20] J. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, Germany, 1984, 1987.

[21] V. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross. *Heterogenous Active Agents*. MIT-Press, 2000.

[22] B. Thomas, Y. Shoham, A. Schwartz, and S. Kraus. Preliminary Thoughts on an Agent Description Language. *International Journal of Intelligent Systems*, 6(5):497–508, August 1991.