Probabilistic Planning in AgentSpeak using the POMDP framework

Kim Bauters[1], Kevin McAreavey[1], Jun Hong[1], Yingke Chen[1], Weiru Liu[1], Lluís Godo[2,1], and Carles Sierra[2,1]

[1] Queen's University Belfast (QUB), Belfast, United Kingdom
[2] IIIA, CSIC, Bellaterra, Spain

Abstract. AgentSpeak is a logic-based programming language, based on the Belief-Desire-Intention paradigm, suitable for building complex agent-based systems. To limit the computational complexity, agents in AgentSpeak rely on a plan library to reduce the planning problem to the much simpler problem of plan selection. However, such a plan library is often inadequate when an agent is situated in an uncertain environment. In this work, we propose the AgentSpeak$^+$ framework, which extends AgentSpeak with a mechanism for probabilistic planning. The beliefs of an AgentSpeak$^+$ agent are represented using epistemic states to allow an agent to reason about its uncertain observations and the uncertain effects of its actions. Each epistemic state consists of a POMDP, used to encode the agent's knowledge of the environment, and its associated probability distribution (or belief state). In addition, the POMDP is used to select the optimal actions for achieving a given goal, even when faced with uncertainty.

1 Introduction

Using the Belief-Desire-Intention (BDI) agent architecture [20], we can develop complex systems by treating the various system components as autonomous and interactive agents [12]. The beliefs determine the desires that are achievable, the desires are the goals an agent wants to achieve and the intentions are those desires the agent is acting upon. A number of successful agent-oriented programming languages have been developed based on this architecture, such as AgentSpeak [19] and Can [21]. Notable BDI implementations include, for example, JASON [5] and JADEX [6]. The benefits of the BDI model in scalability, autonomy and intelligence have been illustrated in various application domains such as power engineering [15] and control systems [12]. Key to the efficiency of BDI agents is the use of a set of pre-defined plans, which simplify the planning problem to an easier plan selection problem. However, obtaining a plan library that can cope with every possible situation requires adequate domain knowledge. This knowledge is not always available, particularly when dealing with uncertain situations. As such, when faced with uncertainty, an autonomous and intelligent agent should resort to other forms of planning to make rational decisions.

To illustrate the problem, consider the example shown in Figure 1. A truck needs to collect materials from three different factories, each producing a distinct type of material that may or may not be available (*i.e.* the environment is *stochastic*). The truck needs to collect all materials by visiting each factory while limiting costs (*e.g.* fuel). The truck agent is uncertain as to whether the material in a factory is ready to collect, but it can use previous experience to estimate a degree of belief. To further complicate the situation, the truck agent can only infer its location by observing nearby signposts (*e.g.* the agent is near a supermarket or a petrol station). Travelling between factories may also fail (*i.e.* non-deterministic actions).

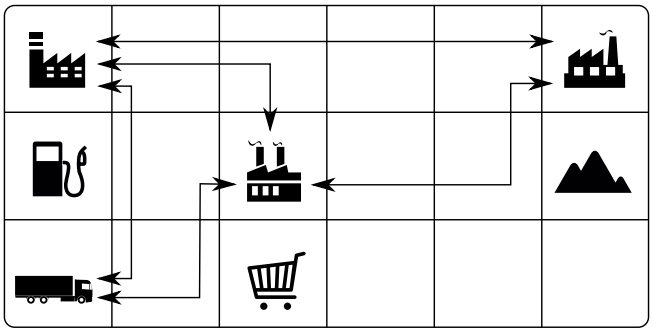

Fig. 1. The material collection scenario.

The large number of possibilities make a pre-defined plan library infeasible, even in this small example. We address these issues by combining AgentSpeak with Partially Observable Markov Decision Processes (POMDPs). POMDPs are a framework for probabilistic planning [13], and are often used as a decision theory model for agent decision making. Other frameworks, such as probabilistic graphplan [4], only consider the *uncertain effects* of actions. Similarly, the *partial observability* of the *stochastic* environment is not addressed by approaches such as probabilistic Hierarchical Task Networks (HTN) [17] and Markov Decision Processes (MDPs) [3]. As such, POMDPs seem to offer an elegant solution to deal with examples such as the one discussed above. In particular, when optimal solutions are required (*e.g.* the truck wants to collect as many materials as possible subject to the fuel limit), POMDPs can be used to compute these solutions. However, even though efficient algorithms to compute the optimal policy exist (*e.g.* [11]), POMDPs are still computationally expensive.

By integrating POMDPs into a BDI architecture, we retain the scalability of the BDI architecture, while adding to it the ability to model uncertainty as well as on-demand access to optimal actions (provided by the POMDP component). The framework we propose is called AgentSpeak$^+$. In this framework we introduce the concept of epistemic states [14] to the BDI paradigm. These epistemic states are used to model the beliefs of an agent about uncertain infor-

mation, along with information on how these beliefs evolve over time. To achieve this, POMDPs are embedded into the agent's epistemic states and are used to represent some aspects of the agent's domain knowledge. The optimal actions generated by these POMDPs can be fed into the agent's plan execution. Therefore, alongside the traditional trigger-response mechanism based on pre-defined plans in BDI, an AgentSpeak$^+$ agent also has the ability to take optimal actions when dealing with uncertain and partially observable environments.

The main contributions of this work are as follows. First, we extend the belief base of a BDI agent with epistemic states consisting of POMDPs to allow an agent to reason about both the partially observable stochastic environment and the uncertain effects of its actions. Second, we present how the agent can delegate POMDPs to find optimal action(s) when the agent is dynamically generating its plans under uncertainty for achieving its goals. Finally, we demonstrate through a scenario discussion how the proposed framework can be used to design agents that are aware of the uncertainty in the environment and are able to react accordingly.

The remainder of our work is organised as follows. Preliminary notions on AgentSpeak and POMDPs are mentioned in Section 2. In Section 3 we propose the AgentSpeak$^+$ architecture which integrates POMDPs into AgentSpeak. A scenario is discussed in Section 4. Related work is discussed in Section 5 and in Section 6 we conclude our work.

2 Preliminaries

We start with some preliminaries on AgentSpeak (see Section 2.1) and Partially Observable Markov Decision Processes (POMDP) (see Section 2.2).

2.1 AgentSpeak

We first define how an agent program can be written. We use $\mathcal{S}$ to denote a finite set of symbols for predicates, actions, and constants, and $\mathcal{V}$ to denote a set of variables. Following convention in logic programming, elements from $\mathcal{S}$ are written using lowercase letters and elements from $\mathcal{V}$ using uppercase letters. We use the standard first-order logic definition of a term[3] and we use $\mathbf{t}$ as a compact notation for $t_1, \ldots, t_n$, *i.e.* a vector of terms. We have [19]:

Definition 1. *If b is a predicate symbol, and $\boldsymbol{t}$ are terms, then $b(\boldsymbol{t})$ is a* belief atom. *If $b(\boldsymbol{t})$ and $c(\boldsymbol{s})$ are belief atoms, then $b(\boldsymbol{t})$, $\neg b(\boldsymbol{t})$, and $b(\boldsymbol{t}) \wedge c(\boldsymbol{s})$ are* beliefs.

Definition 2. *If $g(\boldsymbol{t})$ is a belief atom, then $!g(\boldsymbol{t})$ and $?g(\boldsymbol{t})$ are goals with $!g(\boldsymbol{t})$ an* achievement goal *and $?g(\boldsymbol{t})$ a* test goal.

Definition 3. *If $p(\boldsymbol{t})$ is a belief atom or goal, then $+p(\boldsymbol{t})$ and $-p(\boldsymbol{t})$ are* triggering events *with $+$ and $-$ denoting the addition and deletion of a belief/goal, respectively.*

[3] A variable is a term, a constant is a term, and from every n terms $t_1, t_2, \ldots, t_n$ and every n-ary predicate p a new term $p(t_1, t_2, \ldots, t_n)$ can be created.

Definition 4. *If a is an action symbol and $\boldsymbol{t}$ are terms, then $a(\boldsymbol{t})$ is an* action.

Definition 5. *If e is a triggering event, $h_1, \ldots, h_m$ are beliefs and $q_1, \ldots, q_n$ are goals or actions, then $e : h_1 \wedge \ldots \wedge h_m \leftarrow q_1, \ldots, q_n$ is a* plan. *We refer to $h_1 \wedge \ldots \wedge h_m$ as the* context *of the plan and to $q_1, \ldots, q_n$ as the* plan body.

Following these definitions, we can now specify an agent by its belief base BB, its plan library PLib and the action set Act. The belief base of an agent, BB, which is treated as a set of belief atoms, contains the information that the agent has about the environment. The plan library contains those plans that describe how the agent can react to the environment, where plans are triggered by events. Finally, the action set simply describes the primitive actions to which the agent has access.

On the semantic level, the state of an AgentSpeak agent $\mathbb{A}$ can be described by a tuple $\langle \mathsf{BB}, \mathsf{PLib}, \mathsf{E}, \mathsf{Act}, \mathsf{I} \rangle$, with E the event set, I the intention set and BB, Plib and Act as before [19]. The event set and intention set are mainly relevant during the execution of an agent. Intuitively, the event set contains those events that the agent still has to deal with. When an agent reacts to one of these new events e, it selects those plans that have e as the triggering event, *i.e.* the *relevant* plans. We say that a plan is *applicable* when the context of the plan evaluates to true according to the belief base of the agent. For a given event e there may be many applicable plans, one of which is selected and added to an intention. The intention set is a set of intentions that are currently being executed concurrently, *i.e.* those desires that the agent has chosen to pursue. Each intention is a stack of partially executed plans and is itself executed by performing the actions in the body of each plan in the stack. The execution of an intention may change the environment and/or the agent's beliefs. Also, if the execution of an intention results in the generation of new internal events (or subgoals), then additional plans may be added to the stack.

2.2 POMDP

Partially Observable Markov Decision Processes (POMDPs) have gained popularity as a computational model for solving probabilistic planning problems in a *partially observable* and *stochastic* environment (see *e.g.* [13]). They are defined as follows:

Definition 6. *A POMDP $\mathcal{M}$ is a tuple $\mathcal{M} = \langle S, A, \Omega, R, T, O \rangle$ where S, A, and Ω are sets of states, actions, and observations, respectively. Furthermore, $R : S \times A \to \mathbb{R}$ is the reward function, $T : S \times A \to \Delta(S)$ is the transition function and $O : S \times A \to \Delta(\Omega)$ is the observation function. Here, $\Delta(\cdot)$ is the space of probability distributions.*

Instead of knowing the current state exactly, there is a probability distribution over the state space S, called the *belief state*[4] $b(S)$, with $b(s)$ the probability that

[4] Not to be confused with the *belief base* of an agent, which we see later.

the current state is s. When S is clear from the context, we simply write b instead of $b(S)$. In Definition 6, we have that the *Markovian* assumption is encoded in the transition function since the new state depends only on the previous state. Given the belief state b_t at time t, after performing action a and receiving observation o, the new belief state b_{t+1} is obtained using the Bayesian rule:

$$\begin{aligned} b_{t+1}(s) &= P(s \mid b_t, a, o) \\ &= \frac{O(o \mid s, a) \cdot \sum_{s' \in S} T(s \mid s', a) \cdot b_t(s')}{P(o \mid b_t, a)} \end{aligned} \tag{1}$$

where $P(o \mid b_t, a)$ is a normalisation factor obtained by marginalising s out as follows:

$$P(o \mid b_t, a) = \sum_{s \in S} O(o \mid s, a) \cdot \sum_{s' \in S} T(s \mid s', a) \cdot b_t(s').$$

The decision a at horizon t takes into account both the instant reward as well as all possible rewards in future decision horizons (of the POMDP execution). Given a POMDP $\mathcal{M}$, its policy $\pi : \mathfrak{B} \to A$ is a function from the space of belief states (denoted as $\mathfrak{B}$) to the set of actions. The policy provides the optimal action to perform for a given belief state at each decision horizon, *i.e.* it is the action that should be performed in the current belief state to maximise the expected reward.

Definition 7 (Optimal Policy). *Given a POMDP $\mathcal{M}$ with the initial belief state b_1, π^* is an optimal policy over the next H decision horizons if it yields the highest cumulated expected reward value V^*:*

$$V^*(b_1) = \sum_{t=1}^{H} \gamma^{t-1} \cdot R(s, \pi^*(b_t)) \cdot b_t(s)$$

where $b_t(s)$ is updated according to equation (1) and $\gamma \in (0, 1]$ is a discounting factor to ensure that future rewards are lower. Here $\pi^*(b_t)$ is the action determined by policy π^* and the belief state b_t.

A probabilistic planning problem is then defined as the problem of finding the optimal actions for a given POMDP $\mathcal{M}$ and an initial belief state b (where the optimal actions in a POMDP setting are described using a policy π).

3 Integration of AgentSpeak and POMDPs

We now discuss how AgentSpeak and POMDP can be integrated into a single framework. The resulting framework, called AgentSpeak$^+$, allows us to define agents that can perform on-demand planning based on the POMDP to provide optimal decisions in an uncertain environment. We start by introducing the concept of epistemic states to model the uncertain beliefs of an agent. We define an

epistemic state in Section 3.1 as containing both a POMDP $\mathcal{M}$ and a belief state b, where the former encodes the agent's domain knowledge about the partially observable environment and the latter represents the current uncertain information about the states modelled in $\mathcal{M}$. The basic concepts needed for probabilistic planning are introduced in Section 3.2, where we show how a new construct behaving as an action in AgentSpeak allows for the desired on-demand planning. Our new framework, AgentSpeak+ in then introduced in Section 3.3, where it combines both aforementioned ideas with classical AgentSpeak.

3.1 Epistemic States

Normally, a belief base only contains belief atoms with Boolean values. Such an approach is insufficient to reason over the uncertain beliefs of the agent. To overcome this, we extend the the idea of a belief base into the concept of an epistemic state.

Definition 8 (Epistemic states). *Let $\mathcal{M}$ be a POMDP which models the situated partially observable stochastic environment. By definition, $\mathcal{M}$ includes a set of states S. The epistemic state Φ over the state space S is defined as $\Phi = \langle b, \mathcal{M} \rangle$, where $b : S \to [0, 1]$ is a probability distribution over S.*

The POMDP $\mathcal{M}$ defined in the epistemic state Φ represents the knowledge about the uncertain environment. The observations Ω and state space S in $\mathcal{M}$ are subject to the agent $\mathbb{A}$'s belief base. The action set A of the POMDP can contain both primitive actions in agent $\mathbb{A}$'s action set Act and *compound* actions which correspond to goals achievable by executing existing plans. *Compound* actions, which are plans in BDI, can have various outcomes. However, they can be transformed into primitive actions over which POMDP can reason using a translation such as the one proposed in [9]. Indeed, the work in [9] suggests an approach to summarise sceptical results of compound actions as primitive actions. The belief state b quantitatively denotes the degree of certainty about the partial state space as the initial condition for $\mathcal{M}$.

Example 1. Let $\Phi = \langle b, \mathcal{M} \rangle$ be an epistemic state. The state space S in $\mathcal{M}$ is $\{\mathsf{LocA}, \mathsf{LocB}\}$, *i.e.* two possible locations, where we want to be in LocB (as indicated by the reward function). The current belief state is given by $b(\mathsf{LocA}) = 0.6$ and $b(\mathsf{LocB}) = 0.4$, which will change based on the actions performed by the agent (*e.g.* probably in LocB if you move from LocA) and its observations (*e.g.* probably in LocA when you see a supermarket). The POMDP $\mathcal{M}$ encoding the relevant knowledge is graphically illustrated in Figure 2.

Each epistemic state contains all knowledge needed to plan over and reason about a well-defined subset of the environment. Given that each epistemic state has an optimal policy, this optimal policy intuitively encodes a subplan for this subset of the environment. Throughout this work, we assume that a corresponding symbol Φ is available on the syntactic level as well, *i.e.* on the level of an AgentSpeak agent we are able to refer to a specific epistemic state. An executed action and

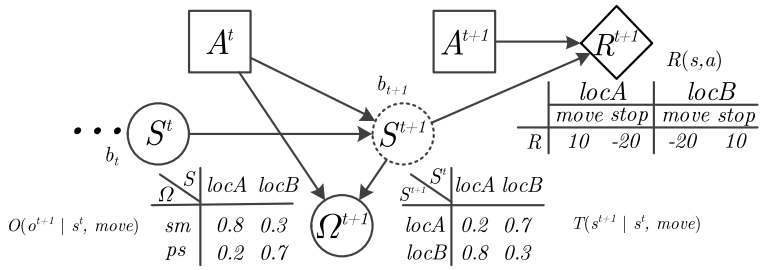

Fig. 2. Graphical representation of the POMDP $\mathcal{M}$. We have that $S = \{\mathsf{LocA}, \mathsf{LocB}\}$, $\Omega = \{(\mathsf{s})\mathsf{uper}(\mathsf{m})\mathsf{arket}, (\mathsf{p})\mathsf{etrol}\ (\mathsf{s})\mathsf{tation}\}$ and $A = \{\mathsf{move}, \mathsf{stop}\}$. The transition, observation and reward functions are shown as tables.

a newly obtained observation are together taken as new input to revise the epistemic state using the belief updating process in Equation (1), where revision is defined as:

Definition 9 (Epistemic state revision). *Let $\Phi = \langle b, \mathcal{M} \rangle$ be an epistemic state and $I = \langle a, o \rangle$ an input with $a \in A$ an action and $o \in \Omega$ an observation. The revision of Φ by I, denoted as $\Phi \circ I$, is defined as:*

$$\Phi \circ I = \langle b', \mathcal{M} \rangle$$

with $\circ$ a revision operator. Particularly, $\circ$ is given by equation (1). The result of revision is a new epistemic state, where the new belief state b' is determined based on the old belief state b and the input I.

Example 2 (Example 1 continued). Let $\Phi = \langle b, \mathcal{M} \rangle$ be the epistemic state from Example 1 with the belief state b. After performing action move, and receiving a new observation ps, the revised probability distribution over possible locations is $b'(\mathsf{LocA}) = 0.16$ and $b'(\mathsf{LocB}) = 0.84$.

It is important to note that revision will only revise the belief state, while keeping the corresponding POMDP unchanged (*i.e.* revision does not alter the domain knowledge of the agent in this case). When there is a sequence of inputs $I_1, \ldots, I_n$ the epistemic state is simply revised iteratively. Furthermore, we assume that an agent can have multiple epistemic states, each dealing with a localised and isolated part of the beliefs of the agent. For example, the availability of material at each factory is independent of the colour of the traffic light. Localised epistemic states allow an agent to revise a corresponding epistemic state given a new input without affecting other epistemic states[5]. This reflects, to some extent, the notion of *minimal change principle* in belief revision.

We will also allow the belief state (*i.e.* the beliefs maintained by a POMDP) to be extracted from the agent's *belief base* (*i.e.* the component of an AgentSpeak

[5] For simplicity, we restrict ourselves in this work to the case where each input is relevant to only one epistemic state.

agent where beliefs are stored). This is useful for designing an AgentSpeak$^+$ agent, as it allows the automatic extraction of the belief state of the POMDP from the AgentSpeak$^+$ program. To simplify the explanation, we will explicitly add the POMDP as a parameter to the belief atoms to make explicit to which POMDP the belief atom is associated.

Definition 10 (Correlated belief atoms). *Two belief atoms $h(x, m, \mathcal{M})$ and $h(x', m', \mathcal{M})$ are said to be correlated if x and x' are two states of variable X_j (which is one of the variables in the joint state space S) defined in the POMDP $\mathcal{M}$, where m and m' are their corresponding probability values.*

Definition 11 (Extraction). *Let $\{h(x_i, m_i, \mathcal{M}) \mid i \in 1, \ldots, k\}$ be a set of exhaustively correlated belief atoms for variable X_j. The belief state $b(x_i) = m_i$ can be directly derived from this set iff $\sum_{i=1}^{k} m_i = 1$ and $X_j = \{x_1, \ldots, x_k\}$.*

Here, a set of exhaustively correlated belief atoms implies that no other belief atoms in the agent belief base are correlated to any of the belief atoms in this set.

When the state space S of a POMDP $\mathcal{M}$ has a set of variables $\{X_1, \ldots, X_n\}$, then the belief state $b(S)$ is the joint probability distribution obtained from $b(X_j)$. When the belief state cannot be extracted from an agent's initial beliefs we assume a default probability distribution, *i.e.* a uniform distribution, for the belief state. Finally, whenever an epistemic state is initiated or revised the belief base of the agent will be updated accordingly using the corresponding triggering events $-h(x_i, m_i, \mathcal{M})$ and $+h(x_i, m_i', \mathcal{M})$ in AgentSpeak.

We can also derive ordinary beliefs from the belief state:

Definition 12 (Derivation). *Let $\Phi = \langle b, \mathcal{M} \rangle$ be an epistemic state containing a probability distribution over S. The belief atom of Φ, denoted as $Bel(\Phi)$, is derived as*

$$Bel(\Phi) = \begin{cases} s_i, \textit{when } P(S = s_i) \geq \delta \\ \mathsf{T}, \qquad \textit{otherwise} \end{cases}$$

Here δ is a pre-defined threshold for accepting that s_i represents the real world concerning S. Notation T is a special constant representing an agent's *ignorance*, *i.e.* an agent is not certain about the state of variable S.

Example 3 (Example 2 continued). The belief state b can be modelled as the belief atoms $\mathsf{location}(\mathsf{LocA}, 0.6, \mathcal{M})$ and $\mathsf{location}(\mathsf{LocB}, 0.4, \mathcal{M})$. The revised belief state b' is represented as $\mathsf{location}(\mathsf{LocA}, 0.16, \mathcal{M})$ and $\mathsf{location}(\mathsf{LocB}, 0.84, \mathcal{M})$.

3.2 Probabilistic Planning

Now that we have defined an epistemic state (which can deal with uncertainty) and how to revise it, we look at how probabilistic planning can be integrated into AgentSpeak. The POMDPs we use in the epistemic state allow us to decide the optimal action at each decision horizon by taking into account the immediate expected reward and the future rewards. However, simply computing the optimal

plan at each step would severely hamper the reactiveness of the AgentSpeak agent due to the computational cost. Instead, we introduce a new action to AgentSpeak, ProbPlan, which can be used in AgentSpeak plans to explicitly compute the optimal action to achieve a goal for a given epistemic state $\mathcal{M}$. This enables the agent to react optimally when needed, *e.g.* for when performing the wrong action likely carries a high penalty. When optimality is not required or when reactiveness is of primary importance, the agent can instead rely on the abstractness and high performance of the normal BDI plan selection strategy.

Definition 13 (Probabilistic planning action). *Let ProbPlan be an ordinary AgentSpeak action symbol and $\Phi = \langle b, \mathcal{M} \rangle$ an epistemic state. We say that ProbPlan(Φ, H) is a* probabilistic planning action*, with H the number of steps and corresponding rewards we should consider (*i.e. *H is the horizon). The effect of executing ProbPlan(Φ, H) is that the probabilistic planning problem defined by a POMDP $\mathcal{M}$ with initial belief state b and horizon H is solved, after which the optimal action $a_i \in A$ is executed.*

Importantly, the action set A defined in a POMDP can contain both primitive actions and *compound* actions (*i.e.* subgoals), each representing different levels of planning granularity. In the latter case, a pre-defined plan in AgentSpeak will be triggered to pursue the goal corresponding to the optimal action a_i. This allows the optimal plan to be as specific as possible (to reach the goal without taking excess steps) while being as abstract as possible (to fully take effect of the domain knowledge encoded in the set of pre-defined plans). The effects of these compound actions can be computed either sceptically (*i.e.* only considering effects shared by all relevant plans) or credulously, which allows us to balance optimality and reactiveness for the given problem. In the first case, we guarantee the outcome of those effects that we want to bring about, but we leave it up to the AgentSpeak reasoning system to select the best plan at time of execution (*i.e.* we are not interested in the side-effects). In the latter case, we apply a "*best effort*" strategy, where we lose some optimality but gain reactiveness. In addition, it should be noted that while the result of ProbPlan(Φ, H) is an optimal action at the time of computation, there is no guarantee that this action will still be optimal during execution. Indeed, the optimal action/-subgoal is not (by default) immediately executed and may be intertwined with the execution of other subgoals which alter the environment. The benefit of not enforcing this optimality but rather *trying* to be optimal is that we retain the reactiveness of BDI and are able to fully use the knowledge already encoded by the system developer in the subgoals.

For the running example, we consider a POMDP with the state space defined as $\{O, A, B, C\}$, *i.e.* the origin location O and three factories A, B and C. In addition, there is an action set consisting of 9 subgoals: 3 subgoals to go from the origin to a factory; and 6 subgoals to go from one factory to another. In all cases, the subgoals consist of both going to the location as well as collecting the corresponding material. For example, we will use goOBcollect to denote the subgoal to move from the origin to factory B in order to collect material B.

Definition 14 (Probabilistic planning plan). *A plan* pl *is called a* probabilistic planning plan *if it contains at least one probabilistic planning action* ProbPlan *in the plan body.*

Similar to Definition 13, a probabilistic planning plan is still a normal AgentSpeak plan. Due to the fact that each probabilistic planning problem defined on $\mathcal{M}$ always has a (not necessarily unique) optimal action, a probabilistic planning plan does not introduce infinite recursion. Furthermore, an optimisation can be applied to reduce the computational cost. Indeed, whenever the first optimal action is decided, a complete optimal policy over H decision horizons has already been constructed as part of the probabilistic planning problem. Before deliberating over the next action, the epistemic state will be revised with the current action and the new observation as defined in Definition 9. Given the revised epistemic state, the next optimal action can then be decided instantly based on the optimal policy that is already constructed without requiring extra computation.

Example 4. Consider the truck agent $\mathbb{A}_t$ from the running example where Φ is the relevant epistemic state. We have:

```
P1: +!collectMaterial : true ← ProbPlan(Φ, 3);
                                ProbPlan(Φ, 2);
                                ProbPlan(Φ, 1).
P2: +!goOAcollect : true ← moveOtoS1; !waitS1toA; moveS1toA;
                           senseLocation; !load(a).
```

The first plan describes how the truck agent can collect all the materials, *i.e.* how it can achieve its goal !collectMaterial. Due to some hard constraint (*e.g.* we only have limited fuel) we rely on the POMDP planning to plan ahead and figure out the best course of action. Since the abstract level considered by the POMDP in the epistemic state Φ can move from one factory to another in a single step, we consider a decision horizon of 3. The result of ProbPlan(Φ, 3) can for example be the subgoal goOAcollect, *i.e.* given all the information available to POMDP at the moment, the optimal action is to first visit factory A. During the execution of this subgoal new observations will be collected (*e.g.* through senseLocation) and will be taken into account when deliberating over ProbPlan(Φ, 2) by (implicitly) using the revised epistemic state to find the optimal action/subgoal to collect the remaining two materials.

3.3 AgentSpeak$^+$

We are now ready to define our AgentSpeak$^+$ framework:

Definition 15 (AgentSpeak$^+$ agent). *An AgentSpeak$^+$ agent* $\mathbb{A}^+$ *is defined as a tuple* $\langle \mathsf{BB}^+, \mathsf{EpS}, \mathsf{PLib}^+, \mathsf{E}, \mathsf{Act}, \mathsf{I} \rangle$*, where the belief base* BB^+ *now contains belief atoms with an associated probability value,* EpS *is a set of epistemic states, the plan library* PLib^+ *contains an additional set of probabilistic planning plans, and* E, Act *and* I *are as before.*

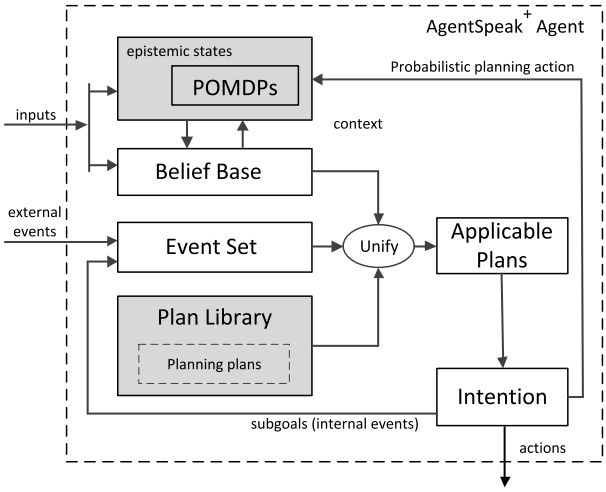

Fig. 3. The revised reasoning cycle for an AgentSpeak$^+$ agent.

Normally, in AgentSpeak, the context of a plan consists of classical belief atoms. However, in AgentSpeak$^+$ the belief base contains uncertain belief atoms, *i.e.* we need a way to determine if a given context is *sufficiently plausible*.

Definition 16 (Belief entailment). *Let* $\{h(x_i, m_i, \mathcal{M}) \mid i \in 1, \ldots, k\}$ *be a set of exhaustively correlated belief atoms in an agent belief base. The belief atom* $h'(x_i)$ *is entailed by the agent's belief base* BB *iff there exists* $h(x_i, m_i, \mathcal{M}) \in \mathsf{BB}$ *such that* $m_i \geq \delta$ *with* $0 < \delta \leq 1$*. The value* δ *is context-dependent, and reflects the degree of uncertainty we are willing to tolerate.*

Example 5 (Example 3 continued). The revised belief base contains the belief atoms $\mathsf{location}(\mathsf{LocA}, 0.16, \mathcal{M})$ and $\mathsf{location}(\mathsf{LocB}, 0.84, \mathcal{M})$. Given a threshold $\delta_{\mathsf{loc}} = 0.8$, only the belief atom $\mathsf{location}(\mathsf{LocB})$ is entailed.

Notice that we can straightforwardly represent classical belief atoms by associating a probability of 1 with them. Verifying if a context is entailed, i.e. a conjunction of belief literals, is done classically based on the entailed belief atoms. As such, we recover classical AgentSpeak entailment of contexts if we enforce that belief entailment is only possible when $\delta = 1$. The revised reasoning cycle of AgentSpeak$^+$ agent is shown in Figure 3. The agent now contains a set of epistemic states, each of which includes a POMDP. A new input can either revise the belief state of a POMDP or be inserted into the agent's belief base BB (*i.e.* this happens when the input is not related to any of the agent's epistemic states). As needed, during plan execution, the agent can furthermore rely on the POMDP to compute the optimal next step through the use of a probabilistic planning action. Whenever the selected plan (*i.e.* the one that has been committed as an intention) contains a probabilistic planning action, the corresponding POMDP will be called instead of (directly) relying on the plan library.

Proposition 1 (Proper extension). *An AgentSpeak$^+$ agent $\mathbb{A}^+$ is a proper extension of a classical AgentSpeak agent $\mathbb{A}$.*

Proof. An AgentSpeak$^+$ agent $\mathbb{A}^+$ extends a classical AgentSpeak agent $\mathbb{A}$ in three aspects. Firstly, an AgentSpeak$^+$ belief base BB^+ extends an AgentSpeak belief base BB by associating a probability value with each belief atom. Secondly, an AgentSpeak$^+$ agent includes a set of epistemic states EpS. Finally, an AgentSpeak$^+$ plan library PLib^+ extends an AgentSpeak plan library PLib by allowing probabilistic planning plans. If all belief atoms in BB^+ have a probability value of 1, if EpS is empty and if PLib^+ has no probabilistic planning plans, then the AgentSpeak$^+$ agent $\mathbb{A}^+$ reduces to a classical AgentSpeak agent. □

Proposition 2 (Termination). *Let PLib^+ be a non-recursive AgentSpeak$^+$ plan library and e an event. If there is a relevant plan for e in PLib^+ then either all steps in the plan body will be executed or the plan will fail in a finite number of steps.*

Proof. If no applicable plan for e exists in PLib^+, then the execution fails immediately. Otherwise, an applicable plan is selected to deal with e and its plan body is executed. If the applicable plan for e in PLib^+ is a classical AgentSpeak plan then the plan body is a finite sequence of (non-recursive) actions/goals. When an action is encountered it is executed immediately. If a goal is encountered, it is either a test goal, which can be executed immediately by querying the belief base of the agent, or it is an achievement goal. If it is an achievement goal, the same line of reasoning we used so far applies for classical AgentSpeak plans. In addition, because we have that PLib^+ is a non-recursive plan library, we know that after a finite number of subgoals (since the plan library is finite) we will have a subgoal for which the plan only contains actions and/or test goals.

If the applicable plan, or any plans of its subgoals is a probabilistic planning plan then the plan body may also include probabilistic planning actions. By definition, a POMDP used by any probabilistic planning action will always return an optimal policy representing a single action/goal to execute. As before, the resulting action or goal will either succeed or fail in a finite number of steps. □

4 Scenario Discussion

We now show how the scenario from the introduction can be expressed using the AgentSpeak$^+$ framework. Since the material collection scenario relies heavily on optimal planning, the scenario serves as a good example to illustrate the benefits offered by AgentSpeak$^+$ over classical AgentSpeak. We stress though that the purpose of this discussion is not to present an actual implementation, but rather to motivate the merits of the proposed framework to warrant future work on a fully implemented system. As a basis for this future work, we briefly discuss a prototype system at the end of this section which we designed to verify the feasibility of our approach.

4.1 Case Study

We recall that the goal of our truck agent $\mathbb{A}_t$ is to collect materials from factories A, B and C. Then, as discussed in Section 3.2, we consider a single epistemic state Φ where its POMDP $\mathcal{M}$ has an action set with 9 subgoals. Each state in the POMDP $\mathcal{M}$ contains information about whether a factory has available materials (denoted as FA, FB and FC), whether our agent has already collected materials from a factory (denoted as MA, MB and MC) as well as the current location of the agent (denoted as LocA, LocB and LocC). For example, the state $s = \{\mathsf{MA}, \mathsf{LocA}, \mathsf{FA}, \mathsf{FB}\}$ indicates that the agent has collected material from factory A, is currently at factory A and that material is still available from factories A and B. We define $\mathcal{M}$ in a graphical way to further decompose the state space (*i.e.* whether material is available at one factory is independent from whether material is available at another). A Dynamic Influence Diagram [1] $\mathcal{D}$ is obtained for efficient computation (shown in Figure 4). In particular, we decompose the entire state space into a set of chance nodes representing FA, FB, FC, MA, MB and MC while LocA, LocB and LocC are represented by a single chance node Loc. Correspondingly, belief atoms in our agent $\mathbb{A}_t$ describe the availability of materials (denoted has(X)), where materials have been loaded from previously (denoted loaded(X)) and the current location of the agent (denoted at(X)) for factories $X \in \{a, b, c\}$. As discussed at the end of Section 3.1, a revision of the epistemic state is triggered when a belief atom is added/removed and possible additions/deletions of belief atoms are triggered when the epistemic state is revised.

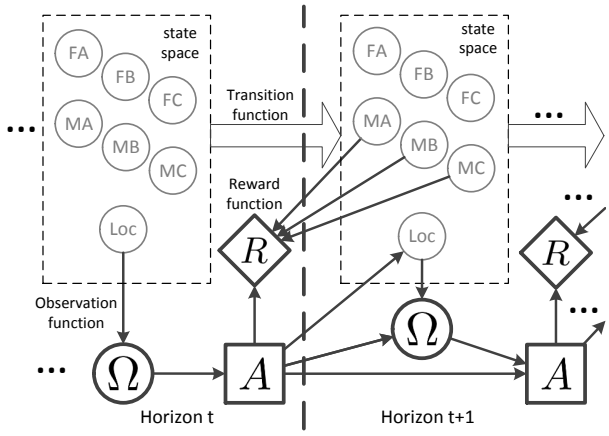

Fig. 4. Graphical representation of POMDP $\mathcal{M}_g$. Here, the entire state space is decomposed into a set of chance nodes. Some causality links for the transition function are omitted for simplicity.

The plan library for our AgentSpeak$^+$ agent $\mathbb{A}_t$ contains many plans, including:

```
(P3)  +!start : true ← !callFactories, !collectMaterial.
(P4)  +!callFactories : true ← !check(a), !check(b),
                               !check(c).
(P5a) +check(X) : not loaded(X) ← call(X).
(P5b) +check(X) : loaded(X).
(P6a) +!waitS1toB : not s1green ← senseSignal; !waitS1toB.
(P6b) +!waitS1toB : s1green.
(P7)  +!load(X) : at(X) ← pay(X); getMaterial(X),
                          !callFactories.
```

We can describe this agent in more detail. When the agent is initialised, the start event is generated. The plan (P3) reacts to this event using plan (P4), along with plan (P5a) or (P5b), by calling each factory to check if material is available (*i.e.* it only calls those factories from which material has not been previously loaded). The agent then proceeds with plan (P3) by attempting to collect the available materials. The material collection procedure itself was previously described by the probabilistic planning plan (P1) from Example 4, which involves a probabilistic planning action using a POMDP to find the optimal plan to execute. A predefined executable plan for moving from the origin location to factory A was described by plan (P2) from the same example. With an equivalent plan for moving to factory B, the plan body would consist of moving the agent to the signal, generating a subgoal relevant to plans (P6a) and (P6b) (*i.e.* to determine when it is safe to pass the signal), and then proceeding to factory B. Once the agent has completed this action it will check whether it has successfully reached factory B. When the location is sensed, and the agent is at the desired factory, it proceeds by executing plan (P7) to load the required material. First the factory is paid, then the material is loaded and, finally, plan (P4) is again executed to call the remaining factories from which material has not been collected. Importantly, the effect of these subgoals (such as moving, loading and calling factories) modify the agent's beliefs and thus also modify the relevant epistemic state. These new beliefs (and uncertainties) are then taken into account by plan (P1) in order to determine the optimal subgoal for collecting the remaining materials.

In addition to a POMDP $\mathcal{M}_g$, the initial belief state b_1 associated with $\mathcal{M}_g$ is also part of an epistemic state. In the preparation stage (relevant to plan (P4)), $\mathbb{A}_t$ defines correlated belief atoms for each variable in the POMDP's state space, such as FA and FB, and these correlated belief atoms are added to the agent's belief base BB. For each variable, such as FA, the corresponding belief state $b(\mathsf{FA})$ can be defined. When this is completed for all variables, a joint belief state $b(S)$ is derived. For example, the action query(FA) estimates the material availability at factory A and belief atoms, such as available(FA, 0.8) and available(FA, 0.2), express our certainty that the material is available at factory A. The beliefs about other factories, the initially loaded material and the starting location of the agent can be obtained in a similar manner. The belief state b_1 itself can be extracted from the relevant belief atoms according to Definition 11.

This example highlights a number of key benefits offered by the AgentSpeak$^+$ framework. Compared to classical AgentSpeak, we are able to deal with uncertain information. Furthermore, our AgentSpeak$^+$ agent is not fully specified in the

design phase; it resorts to probabilistic planning to deal with crucial parts of its execution (*e.g.* determining the order in which to visit the factories). Compared to a pure POMDP implementation, the AgentSpeak$^+$ framework considerably reduces the overall complexity by relying on domain knowledge encoded on the level of a BDI agent. As such, irrelevant actions such as determining which factories to call and how long to wait at a signal, are omitted from the POMDP dealing with the availability of materials. Furthermore, since planning only happens on-demand, the agent can rely on the simpler plan selection process to ensure maximum reactiveness for most of the subgoals (*e.g.* when agents also need to achieve goals where uncertainty is absent and/or optimality is not required).

4.2 Implementation Considerations

A prototype implementation of this framework has been developed[6] that extends Jason [5], an open-source implementation of AgentSpeak, with Hugin [1], a proprietary tool for constructing and evaluating Bayesian networks. Equivalent open source alternatives to Hugin include SMILE [10] and its Java API jSMILE. In addition to this, the implementation uses the flexibility of the Jason system to develop parts of the system in Java. The epistemic states, their revision, extraction and derivation have all been defined on the level of Java. As such, the actual agent description in Jason can be mostly agnostic as to the underlying complexity; beliefs are automatically revised and converted into Boolean beliefs as needed, and the only exposure the agent has to the underlying system is through the ProbPlan concept. Whenever such a ProbPlan action is called, the required POMDP is called through the Hugin API by the Jason interpreter and returns the recommended optimal decision(s). While this prototype proved promising, it suffered from its overall complexity. For example, keeping the beliefs consistent across all three systems is challenging and time-consuming to develop. Still, these tools find optimal solutions for POMDP which can be very time-consuming for even small problems. As a result, the prototype was often considerably slower than plain AgentSpeak. The recent emergency of very capable anytime planning algorithms for POMDP (*e.g.* [23]) is promising and would be the tools of choice for future implementations. Indeed, by using anytime algorithms an agent could further balance between having reactive behaviour, having quick deliberative behaviour or exhibiting behaviour where the agent can wait if it need not act quickly until an optimal solution is found. Such an algorithm could also be integrated in the Java framework, avoiding the need for expensive API calls to an external tool. Finally, we note that a full evaluation of any implementation would require a problem setting considerably larger than the material collection scenario. Indeed, our framework is developed in such a way that planning happens on demand. In realistic scenarios, however, a large part of the environment can be explored without the need for (near-)optimal actions, *i.e.* we can rely on simple plan selection rather than planning based on

[6] By the author Yingke Chen.

POMDPs. For these reasons, the development of a full implementation, as well as its thorough evaluation, is left for future work.

5 Related Work

There have been several approaches to modelling uncertainty in multi-agent systems. In [7] a graded BDI approach is proposed that uses uncertain beliefs (as probabilities) and graded preferences (as expected utilities) to rank plans. However, the theoretical nature of this work has so far precluded the development of any practical implementations. In this work, we instead propose an extension based on a widely used agent-oriented programming language. Our work is based on epistemic states, similar to [8], where the concept of an epistemic state is introduced to model uncertain perceptions. Similar to their work, Boolean belief atoms (*e.g.* propositional statements) can be derived from epistemic states to support further reasoning. Still, the work in [8] only focuses on MDPs, *i.e.* it cannot be used in the more natural setting of partially observable environments. A similar approach has been used in [2] where the authors model different forms of uncertainty as distinct epistemic states. This allows a single agent to reason about different forms of uncertainty in a uniform way. However, the work's main focus is on the representation of the beliefs and their commensurability and does not provide first-principles planning under uncertainty, nor do they exploit any of the facets of the decision theoretical model (*e.g.* rewards or penalties).

Autonomous agents have to make rational decisions to pursue their goals (*i.e.* selecting appropriate plans) in a stochastic environment. Markov Decision Processes (MDP) and Partially Observable MDPs (POMDPs), are popular frameworks to model an agent's decision making processes in stochastic environments. In [22] a theoretical comparison of POMDPs and the BDI architecture identifies a correspondence between desires and intentions, and rewards and policies. The performance and scalability of (PO)MDPs and the BDI architecture are compared in [24]; while (PO)MDPs exhibit better performance when the domain size is small, they do not scale well since the state space grows exponentially. The BDI architecture (which uses a pre-defined plan library) has better scalability at the cost of optimality, making it applicable to significantly larger state spaces. Nevertheless, future research showed that BDI and MDP are closely linked. Indeed, in [25] the relationship between the policies of MDPs and the intentions in the BDI architecture is discussed. In particular, it shows that intentions in the BDI architecture can be mapped to policies in MDPs. This in turn led to some work in the literature on hybrid BDI-POMDP approaches. In [18] an algorithm was proposed to build AgentSpeak plans from optimal POMDP policies. However, most characteristics of the original BDI framework are not retained in such hybrid approaches. In contrast, our approach embeds POMDPs in the traditional BDI agent framework. Normal BDI execution is used by default, with the POMDP component allowing an agent to generate new plans on-demand during execution. Extending the BDI architecture with more elaborate planning techniques has also been investigated in the literature. In [21],

the authors present a formal framework to integrate lookahead planning into BDI, called CanPlan. Similarly, in [16], the authors integrate classical planning problems into the BDI interpreter to allow an agent to respond to unforeseen scenarios. However, neither approach considers issues related to uncertainty.

6 Conclusions

In this work we proposed the AgentSpeak$^+$ in which we extend the belief base of an agent by allowing it to be represented by one or more epistemic states. An essential part of each epistemic state is a POMDP, which allows us to model the domain knowledge of the partially observable environment and from which we can compute optimal actions when needed. In addition, this allows us to deal in a straightforward way with uncertain information in the environment. To keep the computational complexity low, AgentSpeak$^+$ extends AgentSpeak, an agent-programming language based on the BDI paradigm where planning is reduced to the simple task of plan selection. By adding actions to perform on-demand planning, the resulting AgentSpeak$^+$ can both offer good responsiveness while at the same time providing the option for near-optimal planning when needed through the POMDP component. For future work, we plan a full evaluation of our approach, both compared to classical BDI implementations and pure POMDP implementations. We furthermore plan an extension where knowledge from other agents (which are only trusted to some degree) can be employed to improve the domain knowledge currently encoded in the POMDP component.

Acknowledgements

This work has been funded by EPSRC PACES project (Ref: EP/J012149/1).

References

1. Andersen, S.K., Olesen, K.G., Jensen, F.V., Jensen, F.: HUGIN - a shell for building bayesian belief universes for expert systems. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89). pp. 1080–1085 (1989)
2. Bauters, K., Liu, W., Hong, J., Sierra, C., Godo, L.: Can(plan)+: Extending the operational semantics of the BDI architecture to deal with uncertain information. In: Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI'14). pp. 52–61 (2014)
3. Bellman, R.: A markovian decision process. Indiana University Mathematics Journal 6, 679–684 (1957)
4. Blum, A.L., Langford, J.C.: Probabilistic planning in the graphplan framework. In: Recent Advances in AI Planning, pp. 319–332. Springer Berlin Heidelberg (2000)
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-agent Systems in AgentSpeak using Jason. Wiley-Interscience (2007)
6. Braubach, L., Lamersdorf, W., Pokahr, A.: JADEX: Implementing a BDI-infrastructure for JADE agents. EXP – in search of innovation 3(3), 76–85 (2003)

7. Casali, A., Godo, L., Sierra, C.: A graded BDI agent model to represent and reason about preferences. Artificial Intelligence 175(7–8), 1468–1478 (2011)
8. Chen, Y., Hong, J., Liu, W., Godo, L., Sierra, C., Loughlin, M.: Incorporating PGMs into a BDI architecture. In: Proceedings of the 16th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'13). pp. 54–69 (2013)
9. de Silva, L., Sardiña, S., Padgham, L.: First principles planning in BDI systems. In: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'09). pp. 1105–1112 (2009)
10. Druzdzel, M.J.: SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In: Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99). pp. 902–903 (1999)
11. Hansen, E.A.: Solving POMDPs by searching in policy space. In: Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI'98). pp. 211–219 (1998)
12. Jennings, N.R., Bussmann, S.: Agent-based control systems. IEEE Control Systems Magazine 23, 61–74 (2003)
13. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1), 99–134 (1998)
14. Ma, J., Liu, W.: A framework for managing uncertain inputs: An axiomization of rewarding. International Journal of Approximate Reasoning (IJAR) 52(7), 917–934 (2011)
15. McArthur, S.D., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziargyriou, N.D., Ponci, F., Funabashi, T.: Multi-agent systems for power engineering applications – Part I: concepts, approaches, and technical challenges. IEEE Transactions on Power Systems 22(4), 1743–1752 (2007)
16. Meneguzzi, F., Luck, M.: Declarative planning in procedural agent architectures. Expert Systems with Applications 40(16), 6508–6520 (2013)
17. Meneguzzi, F., Tang, Y., Sycara, K., Parsons, S.: On representing planning domains under uncertainty. In: Proceedings of the 3rd Information Theory and Applications Workshop (ITA'10) (2010)
18. Pereira, D., Gonçalves, L., Dimuro, G., Costa, A.: Constructing BDI plans from optimal POMDP policies, with an application to agentspeak programming. In: Proceedings of Conferencia Latinoamerica de Informtica (CLI'08). pp. 240–249 (2008)
19. Rao, A.S.: Agentspeak(l): BDI agents speak out in a logical computable language. In: Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96). pp. 42–55 (1996)
20. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). pp. 439–449 (1992)
21. Sardina, S., Padgham, L.: A BDI agent programming language with failure handling, declarative goals, and planning. Autonomous Agents and Multiagent Systems 23(1), 18–70 (2011)
22. Schut, M., Wooldridge, M., Parsons, S.: On partially observable MDPs and BDI models. In: Proceedings of the UK Workshop on Foundations and Applications of Multi-Agent Systems (UKMAS'02). pp. 243–260 (2002)
23. Silver, D., Veness, J.: Monte-carlo planning in large POMDPs. In: Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS'10). pp. 2164–2172 (2010)

24. Simari, G.I., Parsons, S.D.: On approximating the best decision for an autonomous agent. In: Proceedings of the 6th Workshop on Game Theoretic and Decision Theoretic Agents (GTDT'04). pp. 91–100 (2004)
25. Simari, G.I., Parsons, S.: On the relationship between MDPs and the BDI architecture. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06). pp. 1041–1048 (2006)