Abstract—Achieving faithful representation of knowledge is a historic and still unreached goal in the area of Artificial Intelligence. Computational systems that store knowledge in many different ways have been built in order to emulate the capacity of human knowledge representation, taking into consideration the several inherent difficulties to it. Within this context, this paper proposes an experiment that utilizes two distinct ways of representing knowledge: symbolic, BDI in this case, and probabilistic, Bayesian Networks in this case. In order to develop a proof of concept of this proposal for knowledge representation, examples that will be built through agent oriented programming technology will be used. For that, implementation of a MultiAgent system was developed, extending the Jason framework through the implementation of a plugin called COPA. For the representation of probabilistic knowledge, a Bayesian Network building tool, also adapted to this system, was used. The case studies showed improvement in the management of uncertain knowledge in relation to the building approaches of classic BDI agents, i.e., that do not use probabilistic knowledge.

Keywords—agents, bayesian networks, BDI, multiagent systems, probabilistic knowledge

I. INTRODUCTION

Knowledge representation remains as a major challenge to the community of Artificial Intelligence (AI). According to [1], the exact nature of the role of this area is undefined. Knowledge representation systems can vary from packets to manipulate data structures to complete AI systems that can plan or manage resources.

The main difficulty in representing knowledge as faithful as possible to reality occurs because human knowledge is a very difficult property to be represented in a formal manner which could be interpreted by computer systems in general. Frequently, we have to deal with probabilistic knowledge, on which we rely on incomplete, inaccurate, neutral or even ambiguous information to think and make decisions.

In this paper, we present preliminary cases that seek to integrate probabilistic and symbolic knowledge in the selection of plans for the artificial agents. The chosen framework for the constructions of those agents was Jason. Its architecture system is based in BDI (technology for the knowledge representation) and it is used for MultiAgent System (MAS) programming. Any framework for single-agent or multiple-agent systems could be used, however Jason was chosen because it has all required tools to develop this work.

To facilitate knowledge inclusion in the system, we use a probabilistic network, more specifically, a Bayesian Network (BN), which is a probabilistic graphical model that represents a set of random variables and their conditional dependencies through a directed acyclic graph. Through this representation model, knowledge gains a probability number, i.e., it will no longer be admitted only as true or false, but with a degree of certainty. The BN construction tool used was JAmplia.

The implementation of this work was done through a plugin called COPA. In it, we extended a framework for MAS development (Jason) through reimplementation of the plan selection algorithm and added new features in JAmplia to communicate these tools.

In this article, our main contribution is the insertion of probabilistic knowledge into artificial agents of a BDI system using BN to model the information of the environment. In order to do that, we implemented plugin COPA (Portuguese acronym to Probabilistic Knowledge for BDI Agents).

Besides section I, in which basic concepts for the construction of what is proposed are introduced, this article is divided into five parts. Section II outlines the work related to this paper. In subsequent sections (IV and III), the main tools used are presented in detail, in order to familiarize the reader with such technologies and give him or her necessary knowledge to understand how the developed tool was built. Section V presents the COPA plugin, major changes made in the used tools are also detailed, two case studies are analyzed and the contributions of this work against the classical approach is discussed. Finally, section VI concludes the paper with a discussion about possible future works.

II. RELATED WORK

A model for graded BDI agents was described in [2] which allows the explicit representation of uncertainty beliefs, desires and intentions. The proposed architecture is
based on multi-contexts to model graded beliefs, desires, intentions, plans and communication, in other words, where the grading corresponds to the probability that the agent believes in the context.

Beliefs are tied to probabilities as $B\varphi$, being interpreted as “$\varphi$ is probable” and can be more or less true, depending on the value of $\varphi$.

Intentions are also associated with grading. One can construct a theory $T$ containing the formula $I\psi \rightarrow_L I\varphi$, indicating that the agent should try to run $\varphi$ before $\psi$, and should not try to run $\phi$ if $(I\varphi, \delta)$ is a formula in $T$ and $\delta < \text{Threshold}$. This could mean that the profit to achieve $\phi$ is low or its cost is high.

In [3], a formal framework for practical reasoning based on an argumentative abstract machine is presented. It is divided into three steps: the set of attainable desires is computed given the current state; a set of plans (extensions) that are attainable together is computed; and the results of the two previous steps are combined in order to find the best extension.

BayesJason [4] extends the Jason tool, addressing the integration of BN and BDI agents. To allow BDI agents to act on uncertain environments, BayesJason implements a probabilistic module for Jason. The AgentSpeak(L) grammar was modified and extended, allowing the programming of BN within Jason. Although integrating BN and BDI model, this extension uses the BN in the context of the intention selection (plans in the AgentSpeak(L) systems). BayesJason does not use knowledge representation to verify the compatibility among mental states, neither to bring about states of world where desires can be achieved. Probabilities are used in a limited way, since they are associated with the trigger of the plans instead of the decisions (threshold of each decision). It makes it difficult to explore different behaviors.

The work of [5] describes an approach to self-regulation processes of social exchange based on personality in MAS. Since the agents do not necessarily have access to the internal states of other agents, the decision process should be done in a partially observed way. Therefore, the decision on the best exchange that an agent must propose to another to achieve social equilibrium is modeled as a Partially Observable Markov Decision Process total for each personality trait that the agent can take.

A new algorithm for DEC-POMDPs was developed in [6], which is more robust to model uncertainty.

Despite of the fact that all presented papers manage probabilistic information somehow, only [4] provides the use of BN. The tool developed in [4] does the opposite of COPA. That is, probabilistic beliefs are built by the developer at the base of the agent’s beliefs and can be transformed into a BN, whereas in COPA, agents’ beliefs are mounted in the BN and transported to the agents’ beliefs base. In other words, probabilities are used in the context of intention selection. In COPA, probabilities are used in the context of plan selection.

[2] uses uncertain knowledge in BDI agents in a more general way than this paper. Unlike COPA, probabilities (degrees) are present in all parts of the system, from beliefs, desires, and intentions to plans and communications, even being used to select intentions, as in the COPA.

[3] and [5] did not develop new techniques for uncertain knowledge representation. The first one wanders about Practical Reasoning, taking into account the uncertain factor when deliberating actions to be executed by the agent. The second one developed a technique for social exchanges of agents based on personalities. Since the state of other agents is non-observable by the agent who is proposing a change, it must take into account the various possible states of other agents, creating uncertainty in the process of reasoning.

III. Probabilistic Networks

Uncertain knowledge representation in probabilistic networks allows us to manipulate uncertainty based on founded mathematical principles. Another major advantage compared to other structures of knowledge representation is the inference calculation reduction to various local calculations, using only variables from one node and its neighbours in a graph structure. So, it is not necessary to perform the calculation of the global joint probability distribution. According to [7], its graphical representation makes explicit the dependency relations between variables of the model and it is a powerful tool for system modeling and knowledge acquisition.

A probabilistic model contains two types of information: a qualitative, which is dependency relations, and quantitative, which represents the states of variables and their probability distributions.

A. JAmplia: a Bayesian Network construction tool

Bayesian Network (BN) is a probabilistic graphical model that represents a set of random variables and their conditional dependencies through a directed acyclic graph [8]. Variables can be observable quantities, unknown parameters, assumptions or latent variables, while edges represent conditional dependencies between variables.

JAmplia is an environment where one can create, edit and compile BN. A user can perform various functions of a BN, as moralization, triangulation, identification of clicks, creation of junction trees, creation of evidences, among others. An evidence in a variable is a proposition about the certainty of its states, which affects probabilities of other variables. It may be specific if we are sure about the exact state, or virtual if more than one state can be true.

IV. MultiAgent Systems

MultiAgent Systems (MAS) compose a research area contained in Distributed Artificial Intelligence, which includes issues in distributed computing in AI systems. There are
two main types of MAS: reactive and cognitive. This paper is focused only on the cognitive type. For more detail on the different characteristics of MAS agents, research in existing literature, as [9] and [10], is advised.

Several architectures of deliberative agents are based on the use of three main states of mind: beliefs, desires and intentions, abbreviated as BDI. The idea of applying these concepts to agents originates from the work of Bratman [11] and Rao and Georgeff [12]. This model is based on the explicit representation of the agent’s beliefs (B) – the views that the agent has on the environment, its desires (D) – the preferences of the agent, and intentions (I) – the goals of the agent.

A. The Jason framework

Jason is a framework developed in the Java language for agents programming in writing way in AgentSpeak(XL) language [13]. One of its main uses is the development of MAS. It facilitates the use of a complex term that is strongly associated with beliefs called annotations. It can be used to store any additional knowledge corresponding to that belief. In this paper, we will only add the knowledge of probability related to the belief, identified by the annotation $p$. It will always be associated with a number between 0 and 1, which corresponds to the probability the agent believes a variable has in the current state. Its use is noted by brackets after the belief, as in $\text{rain}(\text{yes}) [p(0.9)]$, representing that the agent believes that the possibility of rain is 90%.

1) How Jason treats uncertainties: Uncertain knowledge is present in all environments, including BDI. Agents often acquire their beliefs, or make perceptions, from not totally reliable sources, or sometimes acquiring partial information, incomplete or ambiguous. To represent this knowledge, agents find themselves forced to make assumptions or working with degrees of truth, assuming that such knowledge is either true or false, not considering anything in the middle of these two classifications.

To show how Jason deals with uncertain knowledge, we use an example based on [14]. This didactic example models the beliefs about the choice that the agent should make: go to the Movies or play Soccer. The choice is based on two aspects: Weather and Players. The agent believes that rainy weather is not a good condition to play soccer, but it is a good call to go to the movies. It also believes that sunny days are good to play soccer, but they were not made to stay in a room watching movies. Finally, our agent believes that soccer games additionally require a group of players. Figure 1 shows the BN constructed to model these beliefs. The tables associated with each node representing their respective Conditional Probabilities Tables (CPT).

Trying to model this example in Jason, the first difficulty is, of course, how to model the probabilistic knowledge, i.e., how to abstract the odds, making the agent work only with certainties. Looking at Soccer node’s CPT, we see that the probability that the agent will play soccer is high only when the weather is sunny and there are enough players. Since we have no information on the status of variables soccer and movies, we must make the decision based solely on the beliefs weather and players. Therefore, we can model the agent as follows:

As this example is purely didactic, the body of the plans (lines 5 and 8) of the agent come down simply to a print command. If the agent believes that the probability of being sunny is high (greater than 70%) and probably there are enough players (greater than 50%), it will find that there are good conditions for playing soccer (line 1 plan). The agent can still choose to play soccer if it is not sure about the rainy weather (below 80%) and is sure there are players (greater than 90%) or even if it is sure it is sunny (greater than 90%) but it is not so sure about players (below 75%). Otherwise, it chooses to go to the movies (line 6 plan), regardless of weather conditions. This model is quite simplistic and abstracted a great part of reality, since the only chance the agent will play soccer is when it believes the weather will be sunny and probably there are enough players. There is the possibility, for example, that the agent chooses to play soccer despite the rainy weather or the insufficient number of players, as proposed in the BN of the Figure 1.

This implementation presents deficiencies, as the fact that the programmer must define the probabilities in the context of intention selection, using a threshold value. In addition, we must use an extremely large logical expression (lines 3-5).

The difficulty of Jason in environments where uncertainty is present is evident on this example. Even though one can simulate probabilistic reasoning, other obstacles were noted,
as the use of thresholds and very long contexts construction.

V. DEVELOPING COPA PLUGIN

The COPA was developed as a plugin, therefore it can be easily attached by a programmer to Jason environment, maintaining compatibility with legacy code. The tool used in this work for agents’ communication was JADE [15], which is an open source environment for developing applications based on agents as FIPA specifications for interoperability between MAS.

A. Modifying JAmplia

When compiling the BN, JAmplia performs all the steps necessary for the compilation of the network. Therefore, a new function was invoked at this point, receiving as parameters the list of variables from the BN. This method contains basically four steps. The first step validates syntactically the name of the node and its states. Then, the probability associated with this variable is retrieved. The third step comprises the variable name, its state and its probability in the form of a belief in the AgentSpeak language, while the last step sends the messages.

JAmplia will no longer behave as a standalone program, but as an JADE agent, so that it can send messages containing the probability of each belief to agents built on Jason.

Even with these modifications, JAmplia users can still use it as before. To run it as an agent, Jade class must be executed, passing as parameter which platform the agent will join, the class which implements it, among other settings.

When the user selects the option of distributing evidence, JAmplia sends a message to all known agents, containing a list of all agent’s belief base literals (from belief_1 to belief_n), attached to their respective probability annotation for each state (from state 0 to sm), as in AgentSpeak syntax. The message format is as shown below:

beliefs_1(s0)[p(x)],...,belief_n(sm)[p(y)]

B. The Jason Agent Implementations

To use the new features proposed in this paper, we must create an agent that extends the default Jason Agent class, to be used by the agent implemented in Jason. This class must contain a selectOption method, responsible for choosing which plan the agent will run, among those available. This method receives as parameter the list of applicable plans, i.e., a list of all plans that the agent believes to be attainable, returning the plan to be executed in the next execution cycle of Jason. Its implementation is described in Figure 3.

If the list received as parameter has only one element, there is no alternative but to choose it (lines 2 and 3). Next, the bbBeliefs list is filled with all agent’s belief base literals that have the annotation p, which indicates the degree of uncertainty that the agent has about it (lines 4-6).

From line 7 to 16, each plan is reviewed. If its context is empty (applicable in any situation), its probability will be equal to the Threshold value. This represents that if no plan has a value greater than the threshold, this plan will be chosen. This variable value does not obey any heuristics, being assigned to 0.25, as the authors of this paper believe to be an average value for the overall odds of plans to be analyzed. Lines 10-13 retrieve the probability value for each belief of the concerned plan’s context, multiplying these values. Then (lines 14-16), considers whether the current plan has the greatest total probability. At the end of the algorithm, it returns the plan with the greatest total probability (line 17).

Note that the comparison symbol of changing the most likely plan at the moment (line 14) is >. Therefore, if two plans have the same probability at the end, the plan to be chosen is the one closest to the top of the list, or the first one to appear in the agent’s source code.

It is also important to note that it is currently assumed that the context of the plan will be made only by connectors “&” (conjunction), using therefore the multiplication of its variables (line 13). Obviously, this fact escapes from reality, as frequently other connectors are used, such as “|” (disjunction) and “~” (denial). For disjunction, the formula \( p(A|B) = p(A) + p(B) - p(A) * p(B) \) should be used, whereas for denial, \( \sim P(A) = 1 - P(A) \) is the appropriate formula. However, these features were not yet implemented in this work.

1) Compatibility with Legacy Code: If no plan has any probabilistic belief, COPA returns the same result as Jason’s original algorithm. As a result, compatibility is maintained with agents that were not initially programmed with the COPA plugin.

This is a very important feature, because the use of this plugin is not advisable in every case, since it is not always possible to model the environment with probabilistic beliefs. Even in the case studies (presented in V-D and V-E), several plans were implemented without probabilistic beliefs in their
context.

C. Creating Internal Actions

Internal Actions are functions that users can build and use in agent code. Fortunately, some of the necessary internal actions had already been implemented by Jason developers in examples that are downloaded along with Jason own software.

The internal actions created were register and addBelief. The first one registers the agent on JADE platform, linking it to a particular service. The second internal action adds beliefs received through the kqml_received plan.

D. Case Study: Movies or Soccer

In this example, the agent must choose which the best option for the moment is: staying at home, playing soccer or going to the movies. The agent source-code is as described below.

```prolog
/*kqml_received(_, _, Belief, _)*/
<- ms.is.addBelief(Belief);
/*activity.*/
<- .print("Stay at home").
/*activity[source{self}]*/
: weather(sunny) & players(yes)
<- .print("Play Soccer").
/*activity[source{self}]*/
: weather(rainy) & players(no)
<- .print("Go to the Movies").
```

Figure 4. Agent source-code of case study Movies or Soccer

The activity goal has three relevant plans (lines 4, 6 and 9). As their initial belief base is empty, the agent has no information regarding beliefs weather or players. Therefore, only line 4 plan is applicable. When the agent receives a message via JADE, the goal kqml_received is added, causing plan line 1 to execute.

We can observe the agent choosing the plan that has the greatest total probability between the beliefs in its context. After the first message that JAmplia sends to Jason agents, they will have three plans on the list of applicable plans. As line 4 plan has no context, it is assigned to Threshold value. It will only be selected for execution if the multiplication of the probabilities of the beliefs weather(rainy) and players(not) or weather(sunny) and players(yes) are higher than the Threshold value.

E. Case Study: Stock market investor agent

The field of economy is full of partial information, incomplete and ambiguous. There are many newspapers, magazines, articles, and other sources of information that, many times, talk about the same news, but with conflicting points of view. Having so much uncertain information to evaluate, and for the fact that a lot of money is at stake, sometimes investors end up taking decisions based not only on reason, but also on emotion. When we take feelings into consideration, not always can we clearly analyze this type of situation, which requests practical and logical reasoning. So, if we are able to build an artificial agent containing all our knowledge on the area, it would certainly make the best decisions, since it would not consider any emotion, maximizing earnings.

In this example, some characteristics of an investor agent and of the economic environment will be represented. So, this example can be applied to any human agent interested in investing in the stock market. The situations addressed in this case study are merely didactic, aiming to illustrate the behavior of the agents where much probabilistic information is present.

1) Case study definition: Supposing that an investment agent has a previously defined amount of money available in the bank, and he must decide if it is more worthwhile to invest part of his capital in the stock market, to withdraw the money that has been invested, or not to perform any transaction. There are two agents: broker and investor. The broker agent receives the information on the present state of economy, calculates the price of shares traded on the day, and forwards this information to the investor. Based on all these variables, this agent decides which plan is more adequate for the moment: investing, withdrawing, or doing nothing.

The investor only makes a decision when it receives the information of new variation in the stock market from the broker. The broker only calculates the daily variation when it receives new information from BN. This information is sent when the BN is compiled, which is executed exclusively by the user.

The only possibility of investment of the agent is the one governed by an index of a fictional stock market. In this case study, the agent’s profit is directly proportional to the variation of this index, i.e., the higher it is, the higher its profits will be.

2) Case study modeling: This case study consists of 2 parts: the BN (the investor agent was modeled with the JAmplia tool) and the investor and broker agents (built in Jason).

For the construction of the BN, characteristics of the stock market were taken into consideration, as well as personality traits of the investors. Probabilities are defined for the future trend of the index of growth of a hypothetical stock market. Although reality presents infinite states, this index is only classified in two ways: good or bad. The present trend of this index is represented by the Current evidence variable. The News variable represents the presence or absence of new important information (almost always considered bad news in the investments context) that might affect the index. The Projection variable the new trend in growth of the stock market index, based in Current and News.
The names for the investor agent’s characteristics are also intuitive in their meaning. The Age variable indicates the age group of the investor, divided into two states: young and old. The Characteristic node represents how the investor normally faces the possibility of new investment, with two possible states: bold and conservative. Based on the values of these two variables, in Money there is an indication of how much the agent is willing to invest: lot or little. If it is little, it might indicate that the agent is willing to withdraw part of the money, instead of investing even more.

Based on Projection and Money, the Investment variable estimates the general trend of investment of the economic agents in the stock market. There are just three classifications: high, medium, and low.

The information on conditional probabilities form the quantitative model of the net and define the “strength” (probability) with which the parent nodes influence the child nodes in the net. The definition of the quantitative model for the investment variable is presented in Table I. The quantitative models for the remaining variables will not be displayed due to lack of space. The values of the probabilities are fictitious and were not based on any specific study, but only on the knowledge of this paper’s authors.

We will start by presenting the broker agent. The only function this agent has is to calculate the stock market quote at a certain moment, based on the beliefs sent by the BN. Besides the support plans, like the one for registration in the JADE environment, and the one for receiving the message, this agent has other five plans, all called calculate, responsible for the calculation of the stock market quote. Each plan represents a state of economy. Multiplying the probabilistic value of each belief for each plan, we get the full probability of the plan. The plan that has the highest total probability will be chosen for being executed.

The five modeled states were Best situation: Current (bad), Projection (good) and News (great); Worst situation: Current (good), Projection (bad) and News (dire); Good situation: Current (good), Projection (good) and News (great); Bad situation: Current (bad), Projection (bad) and News (dire); Undefined situation: it will only be chosen when the variables in the previous plans are not present in the basis of the agent’s beliefs, or still when their total value of probability is smaller than the value of the threshold.

Having these five plans, the one to be chosen will always be the one that has the highest probabilities in general, according to what was explained in section V-B. In the best situation, for example, the variables with the highest probabilities are Current (bad), Projection (good) and News (great), because it is assumed that the agent will have higher profits when the price of stocks are low, but the trend is for steep increase. So, a random number between 0 and 1 is calculated and multiplied by 2. The result of this calculation will be the variation of the stock market. In the worst case modeled, in which the present situation is good, the projection is bad, and news are dire, the same calculation is done, but the result will be negative. Figure 5 shows the implementation of the plan in the best situation.

The investor agent starts its execution with some initial beliefs, as the amount of money that is available, and the value invested at the moment. This agent also receives all the updates of the BN beliefs, but only makes a decision about investment as it receives the message from the broker containing the updated stock market quotes. As it receives this information, the investor decides which plan is more suitable for the moment.

There are five possibilities for action: Investing a lot (if the agent believes that it is and excellent opportunity for high profits, investing 25% of the whole of the amount available); Withdrawing a lot (the agent believes in the fall of the quotes, withdrawing 25%); Investing little (the investor might believe that there is good perspective for growth, but is not certain about it, investing only 10% of the capital); Withdrawing little (there is belief in the fall of the stock market, is not sure about it, withdrawing 10%); Do nothing (it believes that the wisest decision is not to take any action).

3) Executing the case study: As the system starts working, the way agents react in different situations can be analyzed. Each time the BN is compiled, the server agent groups all the beliefs of the net in a list and sends them to the investor and broker agents. As they receive this list, the investor agent adds all the beliefs to its base, while the broker agent calculates the quotes of the stock market in this same state of environment. Finally, the calculation,

<table>
<thead>
<tr>
<th>Money</th>
<th>Projection</th>
<th>Investment</th>
<th>P(Investment)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lot</td>
<td>good</td>
<td>high</td>
<td>0.9</td>
</tr>
<tr>
<td>lot</td>
<td>good</td>
<td>medium</td>
<td>0.1</td>
</tr>
<tr>
<td>lot</td>
<td>good</td>
<td>low</td>
<td>0</td>
</tr>
<tr>
<td>lot</td>
<td>bad</td>
<td>high</td>
<td>0.2</td>
</tr>
<tr>
<td>lot</td>
<td>bad</td>
<td>medium</td>
<td>0.4</td>
</tr>
<tr>
<td>lot</td>
<td>bad</td>
<td>low</td>
<td>0.4</td>
</tr>
<tr>
<td>little</td>
<td>good</td>
<td>high</td>
<td>0.4</td>
</tr>
<tr>
<td>little</td>
<td>good</td>
<td>medium</td>
<td>0.5</td>
</tr>
<tr>
<td>little</td>
<td>good</td>
<td>low</td>
<td>0.1</td>
</tr>
<tr>
<td>little</td>
<td>bad</td>
<td>high</td>
<td>0.1</td>
</tr>
<tr>
<td>little</td>
<td>bad</td>
<td>medium</td>
<td>0.25</td>
</tr>
<tr>
<td>little</td>
<td>bad</td>
<td>low</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Figure 5. Plan in the Best situation for the investor agent
the broker sends the values of the quote to the investor. Having this new information, the investor agent can update its profits (if it has any amount invested), and decide on taking action on its investment.

When the BN JAmlia editor is executed, the probabilistic beliefs can start to be sent to the agents registered in the JADE platform. For that, the Tools option in the menu should be selected, followed by Distribute Evidence. This command executes the following actions on the BN: moralization, triangularization, identification of clicks, and building of a junction tree, resulting in the tree to the left in figure 6. To the right, the probabilistic calculation for each variable for the current state of the environment can be visualized.

F. Comparative

Before asking ourselves what the implementation of the case studies would be like without probabilistic beliefs, it is suitable to question viability. We would have to use probabilities in the context of intentions selection, associating them to the trigger of the plans, as in [4]. This obliges the programmer to define in which situation each plan will be executed beforehand, compromising the search for differentiated behaviors. In order to try to emulate the capacity of COPA in Jason with its present algorithm of plan selection, we reimplemented the case study of the Stock Market Investor Agent. Notes were used in order to define a minimum value that the probability associated to each belief should have for it to become an applicable plan. Figure 7 presents how the best possible situation plan for the investor agent can be implemented.

```
/+*action(source(self))
  : projection{good}[p{P}>0.7] &
  money{lot}[p{M}>0.7] & account{A}[A>(>50)
  :-invested{I};
  :-account{C * 3/4};
  :-invested{I + (C * 1/4)}.
```

Figure 7. Plan that represents the best situation possible for the investor with no probabilistic beliefs

Note that this plan will only be applicable if the probability associated to the lot and bad states of the variables capital and projection, respectively, is above 70%. This fact restricts a lot the possibility of this plan to be executed, since the initial probabilities are rarely above this percentage. In case we decreased this threshold, this plan would probably be applicable to many situations, not giving chance of execution to the other plans (in case they are in the first positions), even the others also being applicable.

VI. CONCLUSION AND FUTURE WORK

In this paper the COPA plugin was presented, which aims to make an exercise of representation of probabilistic knowledge for beliefs of the BDI model. The case studies that were developed used BDI agents modeled in BN. In order to do that, the plan selection algorithm of the agents developed in Jason was altered, using probabilistic belief revision instead of heuristic for choosing the plans to be executed by the agents.

Through the case studies made, it was noted that it was easier to implement the agents, particularly in the development of their plans, because instead of the agent always choosing the first applicable plan from its list, it will now analyze all the possibilities, choosing the one with the highest probability of achieving the goal quickly.

Developed MAS and BM can be united by this plugin, only needing to change the class of the agent that is going to be used for the one developed in COPA, and using the
BN in JAmplia. This new functionality is important in the Artificial Intelligence field since it unites two research areas, making the development of new software possible.

The process for inclusion of probabilistic values obtained from the BN in the BDI beliefs, and the alteration in the plan selection algorithm do not aggregate power for representation of uncertain knowledge in SMA, but it is useful since it makes the development of probabilistic agents easier through the developed capacity of manipulating this type of knowledge.

In section IV-A1, the difficulty found by a Jason programmer when dealing with uncertain information was clear. The case studies presented two situations in which the use of the COPA plugin improved significantly the programming of the agents. Although the case studies are sufficient to prove this improvement, the plugin developed still has to be exhaustively tested in the search for possible improvements and implementation errors.

When users recompile the BN, all beliefs’ probability is sent to the registered agents. As future work, the BN could send only the beliefs’ probability that have been modified, thereby reducing data traffic on the network.

The calculation for total probability of the plans has to be improved, since nowadays we assume that only the operator “&” (conjunction) is used in the plans context, fact that does not correspond to reality. Using the other operators “|” (disjunction) and “~” (negation) require the improvement of this algorithm, in order to calculate the total probability of the plan, evaluating adequately the logical expression of the respective context.

The algorithm implemented in the method selectOption also has to be refined. In the present, its complexity is relatively high, because it compares the beliefs of each plan to all the beliefs on the basis of the agent.

In addition to link probability to beliefs, we can also draw a probability number to agent’s desires. If a desire is often picked and have not had satisfactory results, the agent could decrease its probability, allowing other wishes to be executed as well.

Another new interesting feature is to allow agents to change the odds of its beliefs, sending the new value back to the BN. Thus, the BN user can have real-time view of environmental conditions where agents lie.

REFERENCES


