Graded Belief Revision for Jason: A Rule-Based Approach

Dima El Zein
CNRS, I3S, UMR 7271
Université Côte d'Azur
France
elzein@i3s.unice.fr

Célia da Costa Pereira
CNRS, I3S, UMR 7271
Université Côte d'Azur
France
Celia.DA-COSTA-PEREIRA@univ-cotedazur.fr

***Abstract*—Jason is a Java-based platform for the development of multi-agent systems, which is a particular implementation of AgentSpeak. While some theoretical proposals have been put forward to add to Jason both belief revision and the preference order on the agent's beliefs, the reasoning on the practical way to integrate such proposals as well as their implementation have not been considered. This paper aims to fill those gaps, by adding the concept of graded beliefs and making use of Jason customisation features to implement reasoning and belief revision capabilities. The resulting approach allows agents to reason about the belief's degree of certainty, track dependency between them, and revise the belief set accordingly. A running example will illustrate the presented work and highlight its added value.**

***Index Terms*—Belief Revision, Graded beliefs, Rule-Based agents, Jason.**

I. Introduction and Related Work

Belief revision [1] is the process of handling the agent's beliefs when a new piece of information that is inconsistent with the current beliefs is added. There are two main approaches to belief revision: The *AGM* (Alchourrón, Gärdenfors, and Makinson) style [2] studies how an ideal rational agent should revise its beliefs, with minimal changes, when receiving new information; and the *Reason-maintenance* belief revision style [3], considers tracking dependencies between beliefs.

Belief in the real world is a graded concept: an agent might have beliefs more entrenched than others or ones that are more reliable than others [4]. The degree of a belief can be seen as a degree of preference on beliefs that can be considered during the revision process; it will decide which beliefs have the preference to be kept and which should be contracted to maintain consistency.

We focus in this paper on the Jason language [5]: a well known logic-based agent oriented language that was extended from AgentSpeak(L) [6] and inspired by the BDI (Belief - Desire-Intentions) architecture [7]. One of the most interesting characteristics of Jason is that it is developed in Java and provides the ability to customise some of its functions. It has been extended with several features that were made publicly available [1]. Belief revision however is not performed by the available version(s), which makes the agents potentially believe in contradictory beliefs. Some promising works proposing algorithms for contracting and revising beliefs in Jason exist in the literature. Alechina *et al.* proposed in [8]–[10] some efficient algorithms to be incorporated into the Jason AgentSpeak interpreter, which perform belief revision in polynomial-time. However, as it has been confirmed by the authors themselves, this implementation is not yet available.

When it comes to belief preference order in Jason, the extension in [9] proposed a preference on beliefs that depends on the source/type of information (preferring *perceived information* over *communicated information*), and on the time when information was added (preferring new information over the older one). However, an explicit manipulation of preference/certainty has not been considered, neither degrees associated with new arrived beliefs. Here also, the proposed algorithms have not been integrated into Jason.

Jason plans can potentially serve the representation of knowledge rules since they are composed of conditions and pre-conditions to execute the actions in the body. However, the existing syntax and interpretation lack flexibility for such representation, especially because of its dependency on the order in which the plans are presented in the library and on the conditions' order in a plan. To our knowledge, there was no related work in the literature that adapted the plans to represent knowledge-rules. All of the existing works used the original syntax of plans to this aim and this may not be adapted to reason with beliefs. One of the goals behind this paper is to adapt the plans' syntax to allow the representation of knowledge-rules.

The aim of this work is twofold: (i) to propose a rule-based approach allowing dealing and reasoning with the agent's graded beliefs; (ii) to implement into Jason the new extended belief revision approach supported by existing approaches in the literature — Jason's language lacks the necessary flexibility to represent both rule-based reasoning and graded beliefs.

The proposed implementation allows Jason agents to represent the following example of an agent walking outdoors. The agent's belief base includes two facts belief and one knowledge-rule: it believes that it will be *cold* and *rainy* today with the respective preference order on the beliefs of 0.8 and 0.6 (on a scale of 0 to 1); and also that *when it is rainy and cold, it has then to open its umbrella*. Knowledge-

[1] *http://jason.sourceforge.net/wp/description/*

rules allow the agent to derive new beliefs or goals (i.e. *open umbrella*) using its existing knowledge/beliefs, and by that, to take decisions/actions. The fact that the umbrella is broken is added to the agent's beliefs with the highest preference order, 1. Supposing the agent has another knowledge-rule that states *not to open the umbrella if it is broken*. It should decide whether to open the umbrella or not. At last, it ends up not opening it because the belief justifying *not to open umbrella* (*umbrella is broken*) has a higher preference than *cold* and *rainy* (justifying the *open umbrella*). Unlike what has been proposed in the literature in [8], when the decision of not taking the umbrella is taken, our proposal does not remove the facts that it is rainy and it is cold .

Jensen *et al.* proposed in [11] a non-graded plan-based approach showing practical uses of Nguyen *et al.*'s algorithm [12] and extending Alechina *et al.*'s [10] to allow for the revision of plans as well, not just beliefs. Besides the fact that they do not take into account degrees of belief, we are not convinced by their proposed non-graded approach. If we apply their method to our mentioned example, the rule *when it is rainy and cold, it has then to open its umbrella* would be removed to make the decision/belief of taking the umbrella underivable. We do not agree with that because if it turns out later that the umbrella got fixed, the agent will end up not taking the working umbrella although it is rainy and cold. What we propose here is a graded-based-approach that ensures, without removing rules, that less certain beliefs cannot be derived if their consequence contradicts with more certain beliefs. Biga *et al.* proposed in [13] *G-Jason*, an extension of Jason's inspired by the framework developed by Casali *et al.* [14]. The extension proposes a graded representation for BDI agents that allows reasoning under uncertainty using graded mental attitudes. While their aimed to prioritise the execution of plans, ours is to use "priorities" to maintain belief consistency.

The remainder of this paper is organised as follows: In Section II we present the two belief revision styles existing in the literature. Section III briefly presents the main features of rule-based agents. Section IV summarises the linear-time belief revision and contraction algorithms proposed by Alechina *et al.*. Section V presents Jason, its current limitations, our motivations to extend it, and the section ends with a presentation of Alechina *et al.*'s proposals to implement Jason. In Section VI we propose our extended framework, and finally, some conclusions and perspectives of future work are presented in Section VII.

II. Belief Revision

Belief revision is, by definition, the process of modifying the belief base to maintain its consistency whenever new information becomes available. The AGM belief revision theory [2] defines postulates that a rational agent should satisfy when performing belief revision. In such a theory, a belief base is closed under logical consequence. We consider a belief base K and a new piece of information α. K is inconsistent, when both α and $\neg\alpha$ are in $Cn(K)$, or $Cn(K) = \bot$, or both α and $\neg\alpha$ are logical consequences of K. Three operators are considered: *Expansion* $K + \alpha$: adds a new belief α that does not contradict with the existing beliefs. *Contraction* $K \div \alpha$: removes a belief α and all other beliefs that logically imply/entail it. *Revision* $K * \alpha$: adds a belief α as long as it does not cause a contradiction in K. If the addition will cause inconsistencies in K, the revision operation starts by minimal changes in K to make it consistent with α, then adds α. In particular, if the agent has to contract a belief α, it does not contract other beliefs that derived α , as long they are consistent with the remaining beliefs (*minimal change*) – *coherence approach* [1].

The AGM Postulates for Contraction: The contraction of a formulae α from K should result in a maximal belief base which does not imply α. Let us consider $\div$ the contraction operator. Given two formulas α and ψ, it must have the following properties:

(K$\div$1) The result of a contraction, $K \div \alpha$, is a theory (*Closure*)
(K$\div$2) $K \div \alpha \subseteq K$ (*Inclusion*)
(K$\div$3) If $\alpha \notin K$ then $K \div \alpha = K$ (*Vacuity*)
(K$\div$4) If $\nvdash \alpha$ then $\alpha \notin K \div \alpha$ (*Success*)
(K$\div$5) If $\alpha \in K$ then $K \subseteq (K \div \alpha) + \alpha$ (*Recovery*)
(K$\div$6) If $\alpha \equiv \psi$ then $K \div \alpha = K \div \psi$ (*Extensionality*)

$K\div$1 ensures that the result of a contraction is a belief base (theory); $K\div$2 ensures that there is no new information added in the belief base after contraction; $K\div$3 ensures that contracting a piece of information that is not believed will not cause changes in the belief base; $K\div$4 ensures the success of contraction which would not work if the piece of information to be contracted was a tautology; $K\div$5 ensures that if α is contracted from K then expanding K with α restores K; $K\div$6 ensures that the result of a contraction is syntax-independent.

On another side, the Reason-maintenance belief revision approach [3], considers tracking dependencies between beliefs, so that the reason(s) for believing in a belief α can be traced. When α should be given up, the agent must ensure that α is no longer derivable and give up believing the things that derived it.

III. Rule-Based agents

A Rule-based agent [11] has a belief base consisting of rules (Horn clauses) and facts (ground literals). The facts can originate from different sources and might change over time as a result of the inference process itself or of the addition and deletion of other facts from the agent's belief base. To ensure the consistency of the base after the arrival of contradictory information, a strategy for revising beliefs is necessary: the agent needs to have a way of removing enough facts from its belief base to make sure that a contradiction is no longer derivable.

As in [10], [11] for example, we assume that the agent's beliefs are represented in predicate logic, in the form of literals and Horn clause rules. We fix a set of predicate symbols P, a set of variables X, and a set of constants D. A literal α is a predicate symbol of n arguments followed by n variables or

constants and possibly preceded by a negation symbol $\neg$.
We consider an agent with a finite set R of rules, which are of the form $\alpha_1 \& \alpha_2, \ldots, \& \alpha_n \rightarrow \beta$ where $\alpha_1, \alpha_2, \ldots, \alpha_n (n \geq 1), \beta$ are literals. β is called the derived belief, and each α_i is a premise of the rule. The $\&$ represents the logical *and* operator.

IV. Alechina's Belief Revision and Contraction

Alechina *et al.* [8] proposed belief revision and contraction operations that are efficient in terms of computation cost (linear time in the size of the agent's knowledge base) and satisfy all the AGM postulates but (K÷5), the recovery postulate. They considered a resource bounded agent of a finite state and a finite program that consists of a fixed number of rules used to derive new beliefs from the agent's existing beliefs. To overcome the complexity, they weakened the language and the logic of the agent. The approach associated a preference order (similar to Williams' approach [15]) for each belief and tracked dependencies between them. For every fired rule instance, a Justification J will record: (i) a belief α, which corresponds to the derived belief and (ii) a *support list*, s, which contains the premises of the rules (contextual beliefs of a plan used to derive α). The dependency information of a belief had the form of two lists: *dependencies* and *justifications*. A *dependencies list* records the justifications of a belief, and a *justifications list* contains all the Justifications where the belief is a member of support. The approach represents the agent's belief base as a directed graph with two types of nodes: *Beliefs* and *Justifications*. A Justification has one outgoing edge to the belief it is a justification for, and an incoming edge from each belief in its support list.

Example 1. Suppose we have four beliefs α, β, γ, and μ, visualised in Fig. 1, and a rule α & $\beta \rightarrow \gamma$. The rule means that if the agent believes in α and β, it will believe in γ. In the graph, Justification J_3 is denoted as (γ,[α, β]); γ is the derived belief and [α, β] is the support list. J_3 is in the dependencies list of γ and in the justifications list both α and β. If γ were also derived from μ, i.e. $\mu \rightarrow \gamma$, then its dependencies list would also include another justification J_5 denoted as (γ,[μ]).

Definition 1: Independent beliefs are beliefs having at least one justification with an empty support list (non-inferential justification). They are usually those in the initial belief base or those perceived from the environment.
If the belief α were the result of an observation, its dependencies list would include a justification $J_2 = (\alpha, [])$ containing an empty support list.

Preference on Beliefs and Quality of justifications: As beliefs are associated with preferences, justifications are associated with qualities. In [8], a quality of a justification is represented by non-negative integers in the range $[0, \ldots, m]$, where m is the maximum size of working memory. The lower the value, the least the quality.

Definition 2: The preference value of a belief α, $p(\alpha)$, is equal to that of its highest quality justification.

$$p(\alpha) = \max\{qual(J_0), \ldots, qual(J_n)\} \tag{1}$$

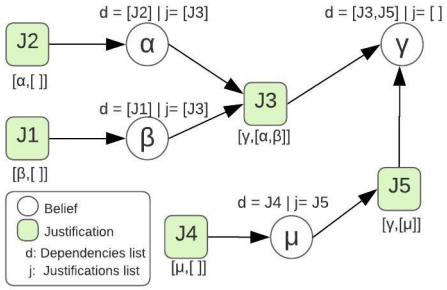

Fig. 1. Graph over the beliefs and justifications.

Definition 3: The quality of justification J, $qual(J)$, is equal to the preference of the least preferred belief in its support list.

$$qual(J) = \min\{p(\alpha) : \alpha \in \textit{support of } J\} \tag{2}$$

With an assumption that non-inferential justification is associated with an *a priori* quality.

The above formulas allow the identification of the weakest member $w(s)$ or "preferred contraction" of a support list s i.e. the member with the smallest preference. The contraction was defined by a literal α as the removal of α and sufficient literals (the least preferred one) so that α is no longer derivable.

Algorithm 1 shows how beliefs are contracted in [8], α is the belief to be contracted, β is a derived belief from α, J is the justification, s is the support list, $w(s)$ is the weakest member of the support list.

Algorithm 1 Contraction by α as per Alechina

```
for each J = (β, s) in α's justifications list do
    remove J from β's dependencies list
    remove J from the justifications list of each literal in s
end
for each J = (α, s) in α's dependencies list do
    if s == [] then
        remove J
    else
        contract by the literal w(s)
    end
end
delete α
```

In the above example, if we want to contract γ, we have to contract μ (since it is the only member of J_5's support list) and the least preferred member of J_3's support list (either α or β) so that γ is not derivable again.

The above algorithm implements the belief revision based on the *coherence theory*. According to the authors, the algorithm can be modified to implement *reason-maintenance* type contraction by removing all beliefs that are not justified anymore (have empty dependencies list).

We consider the knowledge base K and the belief α. The algorithm for revision by Alechina *et al.* is as follows:

Algorithm 2 Revision by α as per Alechina

Add α to K;
apply all matching plans;
while K *contains a pair* $(\beta\ ,\ \neg\beta)$ **do**
  contract by the least preferred member of the pair
end

V. Jason: Its Properties and Limitations– Motivations for our Proposal

We discuss in this section an overview of the Jason language architecture, as well as the current state of the features (in the original Jason v.2.4) we are aiming to extend. We explain throughout some examples the limitations that prevent it from being adapted to the use of rules to represent belief reasoning in particular situations. We finally discuss Alechina *et al.*'s [9] proposed inclusion of automatic belief revision in Jason.

A. Architecture

A Jason agent, similarly to other agents modeled in BDI, is defined by sets of *beliefs*, *goals*, and *plans*.

- Beliefs: represent the information that the agent has currently obtained about its environment. The set of beliefs are represented through a belief base.
- Goals or Intentions: represent the states of affair the agent might like (or has decided) to accomplish.
- Plans: are actions or behaviors defined to accomplish specific goals or intentions. The set of plans available to the agent corresponds to the contents of the agent's plan library.

Jason's beliefs are represented by predicates. Their existence in the belief base express the fact that the agent currently believes that to be true. The $\sim$ operator refers to the negation, which allows the explicit representation that the agent believes a literal to be false.

Annotations distinguish the Jason syntax: an annotation is a list of terms placed after a belief, enclosed in square brackets, revealing details about it. The *source* is the only standard annotation that automatically records the name of the source where the information was obtained. It has a specific meaning and is understandable by the interpreter. Other annotations can be developed by programmers to log the meta-information they want to track about a belief.

A plan is composed of three parts: the *triggering event*, the *context*, and the *body*. It is expressed as follows:

$$+\,triggering\ event : context \leftarrow body. \qquad (3)$$

triggering event represents the condition that might initiate the execution of a plan; it can be the addition or deletion of a belief or a goal; *context* is a conjunction of literals allowing to check if the current situation makes the plan applicable – a plan is chosen for execution if (i) its triggered event occurred and (ii) its context is a logical consequence of the agent's current beliefs; the *body* is a sequence of actions or *goals* to be achieved when the plan is executed. After an occurring event matches the trigger event, the conditions in the context are examined. If they are satisfied, the body of the plan is executed. The syntax of the goals and intentions will not be elaborated, due to their irrelevance to the purpose of this paper.

```
/* Beliefs */
salesUp(company1).
trust(marketstocksite).

/* Plans */
@p1 +salesUp(X)[source(S)] : wellManaged(X)
                & trust(S) <- +goodToBuy(X).
```

Fig. 2. Initial Situation 0

We will propose a new version of the Stock Trader agent example presented in [9], as a running example in the rest of the paper to illustrate the Jason syntax, its properties, and limitations, as well as our new proposed features.

Example 2. We consider a Stock Trader agent that communicates with other agents to receive financial information and has access to Web Services that provide news about the stock market. The agent is trying to decide which stocks to buy or sell, based on its existing beliefs and the information received or perceived.

We present two initial situations for Trader and the related operations to be executed. The operations for each situation are executed sequentially. In the following paragraphs, we will repeat the execution of the same operations and examine the different outcomes in the original Jason version in Subsection V-B and our proposed work in Section VI.

Suppose Trader is represented with the initial beliefs and plans as in Fig. 2. The agent in Situation 0 has two initial beliefs and one plan that will add the belief goodToBuy *when it is executed. Suppose that the belief* wellManaged(company1) *has been added after* ***Operation 0.1****: Agent Ag2 informs Trader in Situation 0 that the* company1 *is wellManaged.*

Situation 1.0 represented in Fig. 3 will be the main example in the paper: The agent believes company1 *is well managed, it trusts the source marketstocksite, the limit order to sell*

```
/* Beliefs */
wellManaged(company1).
trust(marketstocksite).
limits(company2,30).
price(company2,50).

/* Plans */
@p1 +salesUp(X)[source(S)] : wellManaged(X)
                & trust(S) <- +goodToBuy(X).
@p2 +salesUp(X) : ~wellManaged(X)
                          <- +sellStocks(X).
@p3 +salesUp(X) : True <- +watchlist(X).
@p4 +price(X,Z) : limits(X,Y)
                  & Z>Y <- +sellStocks(X).
```

Fig. 3. Situation 1.0

stocks of company2 *is 30 euros and the current price of the company2's stocks is 50 euros. The agent's plan library has four plans, the first three might be executed when the agent is informed that the sales of a company increased, the last one will be executed if the current price of a company's stock has reached or bypassed its limit order.*

When running the agent in Situation 1.0, the interpreter will execute the plan p4 since it is the only one satisfying its conditions, by far. The sellStocks(company2) belief is added into the agent's belief base.

We now describe the operations to execute for Situation 1.0:
Operation 1.1: *Add salesUp(company1): The agent marketstocksite informs Trader that salesUp(company1).*
Operation 1.2: *Add ∼wellManaged(company1): An agent Trader that a crooked CEO has been fired from company1.*
Operation 1.3: *Add ∼trust(marketstocksite): A trusted Web Service broadcasts marketstocksite is not trustworthy anymore.*
Operation 1.4: *Add ∼sellStocks(company2): A trusted Web Service informs holding stock sell for company2.*

We point here that Situations 0 and 1.0 have the same plan p1. The difference is that the agent in Situation 0 believes that the sales of company1 are increasing and then is informed the company is well managed in Operation 0.1; while in Situation 1.0 it already believes the company1 is well managed and then acquires that its sales are increasing in Operation 1.1.

B. Limitations

1) Trigger event and plans' execution: The structure of Jason's plans in Fig. 3 is reliant on the triggering event's occurrence. A plan is executed if the conditions in its context are satisfied before its triggering event takes place. The order of the triggering event and the context conditions matters for the execution of a plan. For example, the plan p1 will not be triggered in **Operation 0.1**; but triggered and executed in **Operation 1.1** resulting in the addition of *goodToBuy(company1)[source(self)]*[2].

Knowing that our aim is to use Jason's plans to represent knowledge-rules; we try to represent *salesUp(X)[source(S)]* & *wellManaged(X)* & *trust(S)* $\rightarrow$ *goodToBuy(X)*: the plan p1 should be replaced by three plans with three triggering events corresponding to the three literals (*salesUp*, *wellManaged*, *trust*).

To represent a knowledge-rule, we would need a number of plans equal to the number of literals in the context +1 (for the trigger condition).

2) Option Selection: Multiple plans in Jason can have the same triggering condition with different contexts. When the triggering event occurs, those plans are set as *Relevant plans*; Then, Jason examines the context of each *Relevant plan* and saves those satisfying their context in a list of *Applicable plans* called *Options*. The *selectOption* function will return one plan only for execution, by default the first option according to the order in which plans were written in the agent code. That restricts the Jason language from representing knowledge-rules using its plan syntax. When two or more rules have the same trigger; and all conditions of the rules are satisfied, only one rule will be fired.

In our Trader example, we would expect both *p1* and *p3* to be executed. Unfortunately, in the available Jason version, it is not possible to represent knowledge-rules in a way to have both plans executed.

3) Belief Base Consistency and Preferences on Beliefs: The default Belief Revision function *brf (Literal belieftoadd, Literal belieftodelete)*, just updates the belief base with the literals to be added or deleted, without checking the belief base consistency. Since contradictory beliefs are accepted in the belief base and no belief revision is performed, there is no preference on beliefs either. The customisation of the *brf* function to include belief revision was left to be done by the programmers.

Going back to our example, the addition of beliefs resulting from **Operations 1.2, 1.3, and 1.4** will be accepted, even if they will lead to inconsistencies in the belief base.

4) Belief certainty: The notion of "believing" in Jason is Boolean: An agent either believes something is true or false or is ignorant about it. The concept of a *belief's Degree Of Certainty* was mentioned by Bordini and colleagues in [16]: An example of an agent Maria believing that another agent Bob was *colorblind* with a certainty of 0.7 was expressed as $colourblind(bob)[source(self), degOfCert(0.7)]$. However, the "degOfCert" in the annotations has no well-defined meaning for the interpreter. Considering such a concept was left to the responsibility of the programmer as well.

C. Alechina's Belief Revision for Jason

Alechina *et al.*'s proposal in [9] consisted of modifying the default *brf* function to include the revision algorithm proposed in [8] and explained in Section IV: The attempt to add a new belief might be discarded or may result in deleting some other beliefs to maintain consistency. They also determined user-specified preference order to decide which beliefs should be deleted to maintain consistency; the preference was given by default to perceived information over communicated; and newer information (in terms of time) over older ones.

While that was one of the first attempts to include automatic belief revision within an interpreter for a practical agent programming language, the authors confirmed it was not yet implemented. The proposed framework is with no doubt promising, as it is efficient and theoretically well-motivated, the reason why we chose to implement and extend it.

VI. A Graded Belief Revision Extension of Jason

We propose an extension to Jason that relies on an efficient approach to revise graded beliefs. Our aim is to use Jason to model rule-based agents, considering beliefs are facts and rules are plans. We also customised Jason plans to allow the representation of knowledge-rules that will derive new beliefs– we introduce the concept of *trigger-independent plans.* Indeed, as we pointed out in Section V-B, the existing approach allows triggering a plan only in the presence of a triggering event. If

[2]Source annotation will be omitted for the rest of the paper.

```
/* Beliefs */
wellManaged(company1)[degOfCert(0.5)].
trust(marketstocksite)[degOfCert(0.8)].
limits(company2,30)[degOfCert(0.6)].
price(company2,50)[degOfCert(0.2)].

/* Plans */
@p1 +tei:salesUp(X)[source(S)] & wellManaged(X)
                    & trust(S) <- +goodToBuy(X).
@p2 +tei:salesUp(X) & ~wellManaged(X)
                                <- +sellStocks(X).
@p3 +tei:salesUp(X) <- +watchlist(X).
@p4 +tei:price(X,Z) & limits(X,Y)
                        & Z>Y <- +sellStocks(X).
```

Fig. 4. Situation 1.0 - New syntax with graded beliefs.

the conditions to execute a plan were completed with a non triggered literal, the plan will not be executed. This is not adapted for dealing with belief change.

One of our goals is to make this triggering process more flexible to allow the representation of knowledge-rules. In addition, we implemented the dependency approach proposed by Alechina *et al.* in [8] and used in [9], to track the dependencies between beliefs by associating dependency and justifications lists for each belief. We adapt their algorithm to calculate the justifications' qualities and the beliefs' preferences (see Equation 1). The result consists of the implementation of a graded notion of beliefs represented by *degreeOfCert* and the proposal of a new algorithm to deal with inconsistency when a new graded belief arrives.

Example 3. We consider, in Fig. 4, a graded version of Situation 1.0 *described in our new proposed syntax. We will execute the same operations performed previously and examine the results when applied to our approach.*

A. Dependency and Certainty

Track dependencies between beliefs, we customised the Jason code to implement the concept of nodes (justifications and beliefs): a justification is represented by a derived belief, a support list, and a quality; a belief is represented by a dependencies list, a justifications list and a degree of Certainty. Whenever a knowledge-rule, named trigger-independent plan, is fired and a new belief is added, the code in Jason is customised to create a justification node that links the context conditions/rule's premises with the derived belief. When any of the beliefs is contracted, the related justifications are over as well. Justifications with empty support lists are created upon the addition of initial, communicated, and perceived beliefs.

We consider that all the sources express their certainty degree, τ, on a belief in the *same scale* $[0, 1]$, i.e a degree τ has the same "importance" for the agent even if it comes from different sources. We do not consider the trust that the agent could have in a source. This is left for future work.

Definition 4: We define the certainty of a belief α as representing the degree to which the agent believes the belief is true. Here, we consider that the preference degree corresponds to the certainty degree, i.e $p(\alpha) = \tau$.

The degree of certainty associated with the*initial beliefs*, *communicated beliefs* and *perceived beliefs* must be explicitly defined by the source (e.g. *wellManaged(company1)[degOfCert(0.5)] is an initial belief defined with a degree of* 0.5). As for derived beliefs their related certainty will be automatically calculated by the interpreter using Equation 1.

As an example, when *salesUp* is added with a degree of certainty 1, and p1 is executed, *goodToBuy(company1)* will be added. The justification for *goodToBuy* will have a quality equal to 0.5 (equal to the least preferred member in its support list = min(0.5,0.8,1)). Therefore, the certainty of *goodToBuy* will be equal to 0.5 since it has only one justification.

Remark 1: Unlike in [8], no *a priori* qualities are assigned for the justification of independent beliefs, as the degrees are explicitly stated.[3]

B. Trigger Independent Plans

We propose an extension to the existing plans' syntax to allow the expression of knowledge-rules. The rules represented by our syntax will be executed whenever the combination of several conditions is true, no matter which condition was satisfied first. We call those rules *Trigger-Independent plans*, as they do not wait for one specific trigger condition to occur to execute the plan. To do so, we needed to specify a plan syntax having a triggering condition that is always true and that is constantly triggered; all the plan conditions will be in the plan's context part. We define "*tei*" that stands for *trigger event independent* as a reserved word for beliefs and make it a part of the Trigger-Independent plans' syntax, as follows:

$$+tei : context' \leftarrow body. \tag{4}$$

$context'$ is composed of the original $trigger\ event$ and $context$ (see for example Plan p1 in Fig. 4).

We modify the *addbel*[4] and *brf* functions to add the "belief" tei with the addition of every belief (initial, communicated, perceived, or derived). As all knowledge rules plans will have *+tei* as a trigger event, they will all be made *relevant plans* upon the addition of any belief. Next, the interpreter will examine the conditions in the contexts, and make those having satisfying contexts as *applicable plans* or *options*.

In our example, when any of the beliefs in the context's condition is added (*trust*, *wellManaged* or *salesUp*), the belief *tei* will be added as well. The plan p1 is triggered, its context is examined (and is valid in this case since all three conditions are there) and therefore executed.

Notice that the users will still have the option to use the original plan's syntax, and alternate between the two syntaxes as needed, i.e. the proposed extension is backward compatible.

[3]We would like to point out that, unlike Alechina's various proposals assuming that only the tautologies can have the highest degree, 1, here, following Dubois and Prade [17], we assume that other formulas are allowed to have the highest certainty. However, they cannot be questioned by revision unless the newly arrived belief also has the highest degree of certainty. In this case, we have chosen the option of keeping the new belief and discard the old ones.

[4]A function in the Agent class that adds beliefs in belief base.

C. Select Options - Multiple plan execution for one trigger

We modified the *selectOption* method to return all the options, one after the other for execution.

Operation 1.1 : Add *salesUp(company1)[degOfCert(1)]*. When *salesUp(company1)* is added, *tei* was added as well by the interpreter. The addition of *tei* will trigger all the knowledge-rule plans. The context of all four plans is examined and the plans p1, p3, and p4 are made *applicable*. The *selectOption* function returns first the plan p1 for execution; *goodToBuy(company1)[degOfCert(0.5)]* is added. The addition of *goodToBuy* will, in its turn, add *tei* again that will trigger all four plans and make them *applicable*. This time *selectOption* will return p3 (the second in the list) since p1 has already been executed; *watchlist(company1)[degOfCert(1)]* is added. The later addition will add *tei* one other time but will not lead to the execution of any plan as all of them have been already executed (p4 was executed when the agent initially ran before adding any operation).

D. Belief Revision

Our contribution here is two-fold: we have modified the *brf* function to check if the addition of a new belief will cause inconsistency in the belief base, and, we integrate the notion of belief's certainty into the belief revision decisions. The belief with the smaller certainty degree in the inconsistency pair will be contracted/discarded, and the other belief will be added/kept. In the case of equal certainties, the new belief will have the preference to be preserved[5].

We propose the following contraction Algorithm:

Algorithm 3 Contraction by α as per our new proposal.

for *each* $J = (\beta, s)$ *in* α*'s justifications list* **do**
  remove J from β's dependencies list
  remove J from the justifications list of each literal in s
  remove J from graph
end
for *each* $J = (\alpha, s)$ *in* α*'s dependency's list* **do**
  remove J from the justifications list of each literal in s
  remove J from graph
end
delete α
Remove all beliefs with empty dependencies list.

Our model does not contract a belief α unless a more preferred contradictory belief $\sim \alpha$ was added. When contracting a belief α, we don't see a need to contract beliefs that derived α: when the rule deriving α will tempt to add it again, the addition will be discarded by the brf function because it was faced by $\sim \alpha$ that is more preferred. In other terms, we contract the belief in question and the related justifications without contracting neither the premises of the rules nor the rule itself.

Remark 2: In comparison with Algorithm 1, we observe that our algorithm does not perform the recursive removal of justifications in the second "for loop". Therefore, the complexity of our algorithm cannot be greater than that of Algorithm 1.

[5]Our implementation gives the developer the flexibility to switch the preference between new and old information.

	Original	**Alechina [8]**	**Our proposal**
Beliefs			
Dependencies	Not tracked	Tracked	Tracked
Inconsistency	Accepted	Not accepted	Not accepted
Graded	No	No	Yes (degOfCert)
Preference[6]	No preference	High Pref $\succ$ Low Pref	High Cert $\succ$ Low Cert; New $\succ$ Old
Belief Contraction			
Contraction of α with $\alpha \rightarrow \beta$	Contract α	Contract α , β	Contract α , β
Contraction of α with $\gamma \rightarrow \alpha$	Contract α	Contract α, γ	Contract α
Plans			
Knowledge-rule with n conditions	n plans	n plans	1 plan
Order of conditions	Dependent	Dependent	Independent (with +tei)
Options execution	1st option only	1st option only	All options

TABLE I
RESULTS OF THE DIFFERENT APPROACHES

Operation 1.2 - Add $\sim$*wellManaged(company1)[degOfCert(0.9)]*. When this operation is received, it will be treated by the *brf* function, that will detect inconsistency with the existing belief *wellManaged(company1)[degOfCert(0.5)]*, and will compare the certainties of the pair. As priority is given to the belief with the higher degree, the *brf* function removes *wellManaged(company1)* first, then adds $\sim$*wellManaged(company1)*. Knowing that wellManaged(company1) was a member of the sole justification of *goodToBuy(company1)*; the contraction of *wellManaged(company1)* will result in the contraction of *goodToBuy(company1)* as well. Finally, p2 is executed as its conditions are satisfied; *sellStocks(company1)[degOfCert(0.9)]* is added.

Operation 1.3 : Add $\sim$*trust(marketstocksite)[degOfCert(0.7)]*. Similarly to Operation 1.2, when inconsistency occurs, the agent prefers to keep the belief with the highest degree. In this case, the addition of $\sim$*trust(marketstocksite)* will be discarded.

Operation 1.4 : Add $\sim$*sellStocks(company2)[degOfCert(0.4)]*. *sellStocks* has a certainty of 0.2 in the belief base. When $\sim$*sellStocks(company2)* is added, it would be more preferred than *sellStocks(company2)*. The agent would then contract *sellStocks(company2)* and add $\sim$*sellStocks(company2)*. On the next reasoning cycle, the plan p4 is made applicable and will attempt to add *sellStocks* with a degree of 0.2 again. However, the addition will be discarded because the agent believes $\sim$*sellStocks* with a degree of 0.4 higher than 0.2. The rule deriving it will be kept, unlike in [18].

We finally summarize our proposal with a comparative table showing the difference between the original Jason, Alechina's approach, and ours.

VII. Conclusion and Future Work

We have proposed a framework that extends Jason to allow revising graded beliefs. To this aim, we have first extended the framework proposed by Alechina *et al.* [8], [9], with the introduction of the concept of *triggering independent plans*, which allows the representation and reasoning of the knowledge-rules. In particular, we have addressed the following issues: (i) The original version of Jason does not perform belief revision and accepts inconsistency in its belief base. (ii) The notion of "believing" is not gradual: An agent either believes something is true, or false or is ignorant about it. While in real life, belief is a gradual concept. (iii) Representing knowledge-rules using Jason plans needs more flexibility as the plan syntax in Jason depends on the order of conditions (satisfaction of the triggering event and the context of the plan) and executes only one applicable plan per triggering event. Therefore, we presented an implemented extension of the Jason language allowing agents to perform belief revision for graded beliefs: We used the belief revision algorithms proposed in [8] for resource-bounded agents which do AGM style belief revision, adapted the work by [9] to Jason, and proposed our own contraction algorithm. We introduced a qualitative order of preference on beliefs as a *Degree of Certainty* representing the extent to which an agent believes a belief is true. This notion of Certainty was integrated with the belief revision and used to decide which beliefs to keep and which to contract in case of inconsistency. An important result is that the complexity of our contraction algorithm is not greater than Alechina's. Finally, our implementation adapted the syntax of plans to accept trigger independent plans, or condition order independent, that would be beneficial for rule-based agents. We compared our proposed approach with the original Jason through a running example illustrating the advantages of our proposal.

As future work, we plan to go ahead improving the representation of the uncertainty related to the agent's beliefs in Jason. Indeed, we are aware that the belief revision process should not be the same for all types of agents. Depending on the type of agents (the reason why it has been programmed for), or on the context in which the agent will behave/act, different adaptations to revision could be suitable. In what we have proposed here, we dealt with a qualitative order on belief preferences. However, we might need to deal with a quantitative order on belief preferences to represent the fact that the agent can change its beliefs only if a "sufficient number" of other trusted sources claim the same contradictory (with respect to the agent's current beliefs) information. In this case, we could adopt the idea of *improvement operators* proposed by Konieczny *et al.* in [19], or the formal model of belief dynamics recently proposed by Hansson in [20], that can be used as another alternative to the primacy of new information followed by AGM belief revision.

Another possible extension could be to take into account confidence in sources, adapting what has been proposed in [21]. In this case, a commensurability hypothesis should be made to address both the certainty about the content of the information, and trust in the source providing the information.

[7] $\succ$ refers to "more preferred".

References

[1] P. Gärdenfors, *Belief revision: An introduction*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992, pp. 1–28.

[2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, "On the logic of theory change: Partial meet contraction and revision functions," *J. Symb. Log.*, vol. 50, no. 2, pp. 510–530, 1985.

[3] J. Doyle, "Truth maintenance systems for problem solving," in *IJCAI*. William Kaufmann, 1977, p. 247.

[4] S. O. Hansson, "Ten philosophical problems in belief revision," *J. Log. Comput.*, vol. 13, no. 1, pp. 37–49, 2003.

[5] R. H. Bordini, J. F. Hübner, and R. Vieira, "Jason and the golden fleece of agent-oriented programming," in *Multi-Agent Programming*, ser. Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005, vol. 15, pp. 3–37.

[6] A. S. Rao, "Agentspeak(l): BDI agents speak out in a logical computable language," in *MAAMAW*, ser. Lecture Notes in Computer Science, vol. 1038. Springer, 1996, pp. 42–55.

[7] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a BDI-architecture," in *Proceedings of KR'91)*, 1991, pp. 473–484.

[8] N. Alechina, M. Jago, and B. Logan, "Resource-bounded belief revision and contraction," in *DALT*, ser. Lecture Notes in Computer Science, vol. 3904. Springer, 2005, pp. 141–154.

[9] N. Alechina, R. H. Bordini, J. F. Hübner, M. Jago, and B. Logan, "Belief revision for agentspeak agents," in *AAMAS*. ACM, 2006, pp. 1288–1290.

[10] N. Alechina, M. Jago, and B. Logan, "Preference-based belief revision for rule-based agents," *Synthese*, vol. 165, no. 2, pp. 159–177, 2008.

[11] A. S. Jensen and J. Villadsen, "Plan-belief revision in jason," in *ICAART (1)*. SciTePress, 2015, pp. 182–189.

[12] H. H. Nguyen, "Belief revision in a fact-rule agent's belief base," in *KES-AMSTA*, ser. Lecture Notes in Computer Science, vol. 5559. Springer, 2009, pp. 120–130.

[13] A. Biga and A. Casali, *G-JASON: An Extension of JASON to Engineer Agents Capable to Reason under Uncertainty*, 01 2014, pp. 17–28.

[14] A. Casali, L. Godo, and C. Sierra, "g-bdi: A graded intensional agent model for practical reasoning," in *MDAI*, ser. Lecture Notes in Computer Science, vol. 5861. Springer, 2009, pp. 5–20.

[15] M. Williams, "Iterated theory base change: A computational model," in *IJCAI*. Morgan Kaufmann, 1995, pp. 1541–1549.

[16] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.

[17] D. Dubois and H. Prade, "Epistemic entrenchment and possibilistic logic," *Artif. Intell.*, vol. 50, no. 2, pp. 223–239, 1991.

[18] J. Spurkeland, J. and J. Villadsen, "Belief revision in the goal agent programming language," *ISRN Artificial Intelligence*, vol. 2013, 2013.

[19] S. Konieczny and R. P. Pérez, "Improvement operators," in *KR*. AAAI Press, 2008, pp. 177–187.

[20] S. O. Hansson, "Revising probabilities and full beliefs," *Journal of Philosophical Logic*, 2020.

[21] V. S. Melo, A. R. Panisson, and R. H. Bordini, "Trust on beliefs: Source, time and expertise," in *TRUST@AAMAS*, ser. CEUR Workshop Proceedings, vol. 1578. CEUR-WS.org, 2016, pp. 31–42.