Plan-Belief Revision in Jason

Jensen, Andreas Schmidt; Villadsen, Jørgen

Published in:
Proceedings of the 7th International Conference on Agents and Artificial Intelligence (ICAART-2015)

Publication date:
2015

Document Version
Peer reviewed version

Plan-belief Revision in Jason

Andreas Schmidt Jensen and Jørgen Villadsen
DTU Compute - Algorithms, Logic and Graphs Section, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, Building 324, DK-2800 Kongens Lyngby, Denmark
{*ascje, jovi*}*@dtu.dk*

Keywords: AgentSpeak, Jason, Plan-belief Revision.

Abstract: When information is shared between agents of unknown reliability, it is possible that their belief bases become inconsistent. In such cases, the belief base must be revised to restore consistency, so that the agent is able to reason. In some cases the inconsistent information may be due to use of incorrect plans. We extend work by Alechina et al. to revise belief bases in which plans can be dynamically added and removed. We present an implementation of the algorithm in the AgentSpeak implementation Jason.

1 INTRODUCTION

Agents receive information from different more or less reliable sources. This information leads to new information by the use of so-called declarative rule plans. When information is shared between many sources it is likely that some of this information is inconsistent, which could result in even more inconsistent information when plans are executed. Furthermore, plans could be wrong, meaning that derived information might not hold. In such situations the agent needs to be able to revise its beliefs to maintain consistency. This process is called *belief revision* and defines three operators: expansion, contraction and revision. The details of the operators will be explained later.

There are two main approaches to belief revision: AGM (short for Alchourron, Gärdenfors and Makinson) style belief revision (Alchourron et al., 1985) and reason-maintenance style belief revision (Doyle, 1977). AGM style belief revision requires that changes to a belief base is as small as possible, where as reason-maintenance style belief revision tracks how beliefs justify each other, using this information to render inconsistent beliefs underivable.

In (Nguyen, 2009) the algorithm for literals was extended to include contraction by rules. Our aim is to show practical uses of the algorithm. Furthermore, our focus is on situations where plans are exchanged or derived.

We consider reason-maintenance belief revision in the implementation of AgentSpeak called Jason. Revision of literals in Jason has been considered before (Alechina et al., 2006a). We show how this can be extended to include revision of plans as well.

The paper is organised as follows. In section 2 we briefly introduce Jason. We motivate the need for revision of plans and beliefs in section 3. In section 4 we describe belief revision for both beliefs and plans. Section 5 describes how belief revision can be implemented in Jason. In section 6 we give an example of how belief revision can maintain consistency in an agent's belief base. Finally we conclude our work by summarising the contribution and introducing directions of future work.

2 JASON

We provide an overview of the Jason interpreter by introducing how to program multi-agent system using it, however we will not go into details with all parts of the system. The overview should give a basis for understanding simple systems using Jason. A thorough description of Jason is found in (Bordini et al., 2007).

The language of Jason, AgentSpeak, is a Prolog-like logic programming language which allows the developer to create a plan library for the agent. A plan in AgentSpeak is basically of the form

```
+triggering event : context <- body.
```

Roughly speaking, if an event matches a trigger, the context is matched with the current state of the agent. If the context matches the current state, the body is executed; otherwise the engine continues to match contexts of plans with the same trigger. If no plan is applicable, the event fails.

The fact that AgentSpeak is a logic programming language allows one to transfer certain specifications written in logical formulas of a multi-agent system to an implementation written in Jason. For instance, part of a plan for a vacuum cleaner agent [9] is shown below:

```
+!cleaning : in(X,Y) & dirt(X,Y)
  <- do(suck).
```

The plan is triggered by the goal `!cleaning`, so if the vacuum cleaner is in a "cleaning state", this triggering event would be applicable. The context specifies that this plan is relevant if the agent currently is somewhere in the environment which is dirty. If the context can be unified with beliefs from the belief base of the agent, it will perform the body, which in this case means that it will perform the action `do(suck)`. However, as mentioned it is quite possible to have several plans for the same triggering event if those plans have different contexts:

```
+!cleaning : in(X,Y) & dirt(X+1,Y)
  <- do(right).
```

This plan will then be applicable if the agent has perceived dirt in an area to the right of its current area. In that case, it will perform the action `do(right)`.

3 CASE STUDY

We present a scenario in which inconsistency arise as a result of both exchange of beliefs and plans. An agent is on a mission to retrieve a treasure. Its main goal is achieved by first finding the treasure and then retrieving it. Finding the treasure can be achieved by conducting a search or by asking other agents for the location. The agent knows that it should dig for the treasure when it has found a plausible location:

```
+at(X) <- +dig(X).
```

Consider the following situation (`#n` refers to the fact that beliefs are revealed over time). We write $Ag_1 \triangleleft$ `+a`$[Ag_2]$ when Ag_1 learns `a` from Ag_2. `~a` denotes strong negation in Jason.

`#1` Ag_1 broadcasts a request for the treasure location.

`#22` $Ag_1 \triangleleft$ `+at(a)`$[Ag_2]$

`#28` $Ag_1 \triangleleft$ `+~at(a)`$[Ag_3]$

`#45` $Ag_1 \triangleleft$ `+~at(b)`$[Ag_3]$

The belief base of Ag_1 is now in an inconsistent state (since it knows both `at(a)` and `~at(a)`). Furthermore, Ag_1 perceives a plausible location for treasure (using its sensors), meaning it both knows `at(b)` and `~at(b)`[1]:

`#35` $Ag_1 \triangleleft$ `+at(b)`$[Ag_1]$

Even though the situations are a bit different (two communicated beliefs versus a communicated belief and a percept), we argue that in each pair, one literal should be removed; the treasure cannot both be and not be at the same location. Intuitively, the agent would want to discard `~at(b)`$[Ag_3]$ since it may not know whether Ag_3 is deceitful, but it trusts its own sensors. However, since both beliefs about location *a* are communicated, it is harder to decide which one to keep. We will elaborate on this later in this paper.

Finally, Ag_4, who does not know the location of the treasure, believes that the treasure is found in a tree, and shares a plan for retrieving it:

`#59` $Ag_1 \triangleleft$ `{ +!get_treasure : at(X) <- +climb_tree(X); +~dig(X) }`$[Ag_4]$

Following this plan could lead to a state where both `dig(b)` and `~dig(b)` are in the belief base.

Clearly some kind of belief revision is needed. We describe an approach which can be used to revise beliefs in the next sections.

4 BELIEF REVISION

We consider two approaches for belief revision: AGM (Alchourron, Gärdenfors and Makinson) style belief revision (Alchourron et al., 1985) and reason-maintenance style belief revision (Doyle, 1977). The AGM approach requires that a revision of a belief base removes as little as possible in order to obtain consistency. This means that even though an agent gives up believing P, it should not give up believing things which were solely justified by P; unless they are inconsistent with the rest of the belief base, in which case they should be revised as well. In reason-maintenance style belief revision, the agent must not only give up P, but ensure that P is no longer derivable from the remaining set of beliefs and rules. This is realized by keeping track of dependencies between beliefs, so that the reason for believing P can be traced back to a set of foundational beliefs. One of these beliefs should then be removed, ensuring that P is no longer derivable from the beliefs in the belief base.

In AGM belief revision the belief base is closed under logical consequence. Three operators are defined: expansion, contraction and revision. *Expansion*, $K + A$, which adds A to K and closes the resulting set under logical consequence. *Contraction*, $K \dot{-} A$

[1] Believing both `at(a)` and `at(b)` could also be considered inconsistent, but that is out of scope for this paper.

(contract by A), removes A from K and changes the set K so that it no longer entails A. *Revision*, $K\dot{+}A$, modifies K to make it consistent with A before adding A. If adding A to K is consistent with K, then the revision operator is equal to the expansion operator.

In general there is no unique maximal set $K' \subset K$ which does not imply A, so contraction and revision cannot be defined uniquely. The AGM theory characterises the set of "rational" contraction and revision operators by a set of postulates (Alchourron et al., 1985). The postulates for contraction are:

- $(K\dot{-}1)$ $K\dot{-}A = Cn(K\dot{-}A)$ (closure)
- $(K\dot{-}2)$ $K\dot{-}A \subseteq K$ (inclusion)
- $(K\dot{-}3)$ If $A \notin K$, then $K\dot{-}A = K$ (vacuity)
- $(K\dot{-}4)$ If not $\vdash A$, then $A \notin K\dot{-}A$ (success)
- $(K\dot{-}5)$ If $A \in K$, then $K \subseteq (K\dot{-}A)+A$ (recovery)
- $(K\dot{-}6)$ If $Cn(A) = Cn(B)$, then $K\dot{-}A = K\dot{-}B$ (equivalence)

where $Cn(K)$ denotes the closure of K under logical consequence.

While the definition looks desirable, it seems to only apply to idealised agents, since it requires that the belief base is closed under logical consequence. In order to be able to revise a belief base which cannot be assumed to be closed under logical consequence, we weaken the logic and language of the agent, so that it corresponds to a typical rule-based agent (Alechina et al., 2006b). This approach leads to a polynomial time algorithm, which satisfies all of the AGM postulates but $K\dot{-}5$, the recovery postulate. The algorithm allows for both AGM and reasoning-maintenance style belief revision.

In (Alechina et al., 2006b) the weakened logic is called W. In the corresponding language, L_W, a well-formed formula is either a literal (P or $\neg P$) or a plan ($P_1 \wedge \cdots \wedge P_n \rightarrow Q$).

The only rule is a generalised modus ponens (GMP):

$$\frac{\delta(P_1), \ldots, \delta(P_n) \qquad P_1 \wedge \cdots \wedge P_n \rightarrow Q}{\delta(Q)}$$

δ is a substitution function replacing all free variables with constants. The agent uses GMP and its belief base to reason. When the rule is fired, the derived (ground) literal $\delta(Q)$ will be justified by $\delta(P_1), \ldots, \delta(P_n)$.

4.1 Justifying Plans and Beliefs

A derived belief has in most previous work only been justified by the beliefs used to derive it, but not the plan that was fired. A reason for this is that the plans of an agent is usually a part of a plan library that is assumed to be correct and unlikely to be modified. However, as described in (Nguyen, 2009), this is not always the case. In fact, even if a derived belief (Q) turns out to be wrong, this should not necessarily mean that any of the supporting beliefs ($P_1, \ldots, P_n$) are wrong; it may just be wrong to conclude Q from them (i.e. the plan is incorrect). In this case it might be better to remove the plan rather than removing P_i to make Q underivable. Otherwise, if the agent realizes that P_i is actually true and the plan was not removed, Q can again be derived.

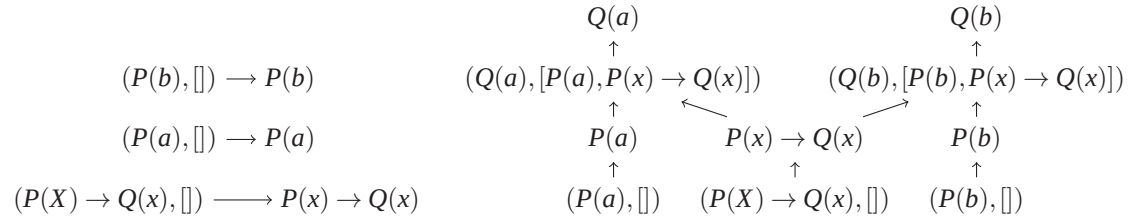
Giving up the assumption that plans are part of a static library and will not be changed, means that plans are now also beliefs. The belief base of an agent will therefore consist of both literals and plans.

We can now define justifications. All formulas are associated with one or more justifications. A justification consists of a *supported formula* and a *support list*. The formula is supported by the formulas in the support list. We write $(A, [B\,C])$, when A is supported by B and C. A formula is independent, if it has a non-inferential justification (i.e. a percept, mental note or communicated belief). In that case, the justification has an empty support list.

Each formula P is associated with a *dependency* and *justification* list. The dependency list for P contains all justifications for P, i.e. all justifications where P is the supported formula. The justification list for P contains all justifications where P is a member of the support list. If we consider the belief base to be a graph, each justification then has exactly one outgoing edge (to the formula it is supporting) and zero or more incoming edges (from the formulas in the support list). Non-inferential justifications will have zero incoming edges.

Example 1. Consider an agent with a plan, $P(x) \rightarrow Q(x)$, and the beliefs $P(a)$ and $P(b)$. The dependency graph in figure 1(a) shows that all formulas in the belief base are non-inferential.

Running the plans of the agent to quiescence (i.e. applying the only plan to both $P(a)$ and $P(b)$) yields two new beliefs, $Q(a)$ and $Q(b)$. This is shown in figure 1(b). If for instance $\neg Q(b)$ is introduced and we choose to contract by $Q(b)$, we can use the relation between formulas to backtrack from $Q(b)$ to the formulas supporting it. □



(a) Initial formulas and dependencies

(b) After firing plans to quiescence

Figure 1: Dependency graph of an agent.

4.2 Contracting by Formulas

When contracting by a formula, we need to further revise the belief base such that the formula is no longer derivable. This requires us to not only remove the justifications for the formula, but also contract by a support from each justification. We use the notion $w(s)$ for the *least preferred* member of a support list s (Alechina et al., 2006b). This formula will be the formula that is not preferred to any other formula in the list, such that we will be prepared to give it up first.

Algorithm 1 is similar to the contraction algorithm given in (Alechina et al., 2006a; Alechina et al., 2006b), with the main difference that a justification can have edges to plans as well as literals. This allows us to track dependencies through plans and thereby contract by them. The check within the first loop corresponds to reason-maintenance style belief-revision. If AGM style belief-revision is preferred, we do not contract by C.

Algorithm 1: Contraction by formula A.

```
for all j = (C, s), where A ∈ s do
    remove j from the graph
    if C has no justifications then
        contract(C)
    end if
end for
for all j = (A, s) do
    if s = [] then
        remove j from the graph
    else
        contract(w(s))
    end if
end for
remove A
```

The overall complexity of the algorithm given in (Alechina et al., 2006b) is $O(kr+n)$ where k is the maximal number of supports in any support list, r is the maximal number of justifications with non-empty supports and n is the number of literals in the belief base.

This does not change when contracting plans as well. The upper bound for the first loop is $r(k+2)$; one constant time operation for the plan itself, one for the formula asserted by the plan, and k operations for the premises of the plan. The second loop has an upper bound of n. The overall complexity is therefore $O(kr+n)$.

In (Alechina et al., 2006b) it was shown that the contraction algorithm satisfies AGM postulates $(K\dot{-}1)$–$(K\dot{-}4)$ and $(K\dot{-}6)$. $(K\dot{-}5)$ is not satisfied, since if B is supported by A and we contract by B, then both A and B are removed. If we add B again, A is not added as well.

4.3 Revision Using Preferred Contractions

We need to elaborate on how to select candidates for contraction. Ideally, the quality of a justified formula should be somehow related to the formulas that supports it (Alechina et al., 2006b). We say that the preference of a formula with the justifications $j_0, \ldots, j_n$ is:

$$p(P) = max\{qual(j_0), \ldots, qual(j_n)\}$$

Furthermore, the quality of a justification $j = (Q, [P_0, \ldots, P_n])$ is the preference value of the least preferred formula in its support list:

$$qual(j) = min\{p(P_0), \ldots, p(P_n)\}$$

We write $j = (Q, [P_0, \ldots, P_n], n)$, when $qual(j) = n$.

It is now possible to order beliefs using a total preference order relation, $\preceq$, on a set of beliefs (Nguyen, 2009). Given two beliefs, P and Q, we write $P \preceq Q$ if Q is preferred over P. This is the case if $p(P) < p(Q)$. Furthermore, if $p(P) = p(Q)$, we can use other measures to determine a strict preference relation, such as taking the age of a belief into consideration.

The quality definition only holds for inferential justifications. It is clear that non-inferential justifications must have some kind of *a priori* quality. In (Nguyen, 2009) the following order of independent

beliefs is suggested:

Percept
> Mental note
> Built-in plan
> Communicated data

As we shall see, choosing different orderings may lead to very different contraction results.

Example 2. Recall the situation from example 1. Suppose we choose to contract by $Q(b)$[2]. All justifications which has $Q(b)$ in their support list are then removed. $Q(b)$ does not support any beliefs, so nothing happens. All justifications for $Q(b)$ must then be removed. There happens to be one: $(Q(b),[P(b),P(x) \rightarrow Q(x)])$.

In order to render $Q(b)$ underivable, one of the supports must be contracted as well. Depending on which formula is preferred least, this gives the two situations shown in figure 2.

We assume that $P(a)$, $P(b)$ are percepts and $P(x) \rightarrow Q(x)$ is a built-in plan (i.e. not communicated from another agent).

In figure 2(a) the literal $P(b)$ has been removed. In this case, the ordering of independent beliefs is Built-in plan > Percept.

Figure 2(b) shows a setting where the plan $P(x) \rightarrow Q(x)$ has been removed. In this case, the ordering of independent beliefs is Percept > Built-in plan. □

The example shows that different preference relations result in very different contraction results. In some cases plans might generally be preferred to literals, while in other cases we might want to prefer communicated beliefs the most. It is difficult to decide when one preference ordering is better than another, since it depends on many factors such as the reliability of the agents and correctness of their plans.

Algorithm 2: Revision by A.

add A to belief base
while belief base contains a pair $(B,\neg B)$ **do**
  $contract(w(B,\neg B))$
end while

Using the preferred contractions, we can now define the revision operation (algorithm 2). In (Alechina et al., 2006b) the revision operator and the basic AGM postulates for it are described. It is furthermore shown that the operator satisfies five of the six postulates. $A \in K\dot{+}A$ is not satisfied since if we add A and use it to derive B, but we already have $\neg B$ in the belief base, then if $p(B) < p(\neg B)$ we would contract by B and finally remove A.

[2]We could do this if, for instance, the agent perceives $\neg Q(b)$.

4.4 Belief Revision in Jason

In Jason plans will not always be of the form required by L_W. When considering belief revision, we therefore only consider so-called declarative rule plans (DRP), which are plans of the form

$$+t : l_1 \,\&\, ... \,\&\, l_n \leftarrow g_1 \,\&\, ... \,\&\, g_m$$

where t is a triggering event, l_1 & ... & l_n is the context, which must be a conjunction of literals and finally the plan-body, which contains one or more belief additions, g_1 & ... & g_m. If the triggering event is a belief addition event, then it will support each g_i together with the context.

Given a DRP:

$$+t : l_1 \,\&\, ... \,\&\, l_n \leftarrow g_1 \,\&\, ... \,\&\, g_m$$

where t is a belief addition event, it is straightforward to show its relation to GMP:

$$(t \wedge l_1 \wedge \ldots l_n \rightarrow g_1) \wedge \cdots \wedge (t \wedge l_1 \wedge \ldots l_n \rightarrow g_m)$$

If t is not a belief addition event it is not included in the antecedent of each implication.

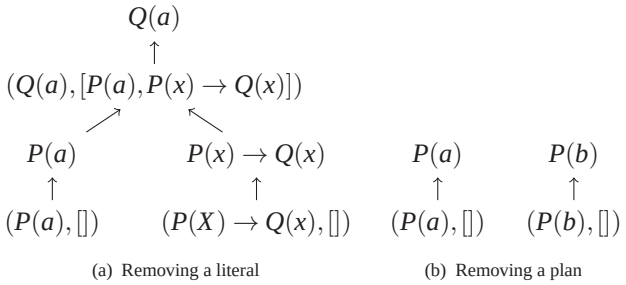
When considering plans we allow g_i to be the definition of a plan, allowing us to add new plans to the belief base. g_i can then either be the addition of a literal ($+a$, $+{\sim}a$) or the addition of a plan, using the internal action `.add_plan`.

If a single g_i is contracted, all other g_j that were derived in the same plan may very well be contracted as well, assuming they are not justified by any other plans. This could lead to situations where having a single contradiction can lead to a much smaller belief base after contraction.

If we contract by g_i, g_i is no longer re-derivable from the set of plans *used so far*. This means that the agent could have a plan $+t : l_1$ & ... & $l_n \leftarrow g_i$ that will not be removed, even though it could be used to derive g_i. This is a drawback of using belief revision in a non-quiescent setting.

5 IMPLEMENTATION IN JASON

We implement belief revision in Jason by extending the **buf** (belief update function) and **brf** (belief revision function) methods of the agent (Alechina et al., 2006a). The belief update function updates the belief base with everything that is currently perceived in the environment and removes anything in the belief base that is no longer perceived. In other words,



(a) Removing a literal (b) Removing a plan

Figure 2: Contracting by $Q(b)$ with different $w(s)$.

it ensures that anything the agent can perceive in the environment can be found in the agents belief base. Since percepts are independent beliefs, we extend the **buf** method to create a non-inferential justification for each percept. The standard implementation of the belief revision function adds and removes beliefs from the belief base. We extend it to revise beliefs using the revision and contraction algorithms when plans marked for revision are executed.

Beliefs and plans are annotated with their dependency and justification lists. Plans that are part of the agents plan library a priori are independent and will have non-inferential justifications. The belief `at(a)`, with a non-inferential justification of quality 3 is annotated as follows:

```
at(a)[dep([j(at(a),[],3)]),just([])]
```

This allows the programmer to create plans that only applies to independent literals, or to literals with a certain preference. For instance

```
+!goal : at(a)[dep(L)]
    & .member(j(_,[],2), L)
  <- ...
```

is only applicable if `at(a)` is an independent belief with $p(\texttt{at(a)}) = 2$.

Communicated plans are – along with communicated beliefs – also independent but have a lower preference. Using the internal action `.add_plan` the agent can add plans dynamically to its plan library. We capture this by extending the *PlanLibrary* so that all plans that are added will be added with appropriate justifications.

Adding a plan does not lead to an iteration of belief revision. Remember that Jason does not run in a quiescent setting, so the plan is not immediately fired. Therefore the plan *could* lead to inconsistency, but this will only be discovered whenever it is actually fired.

We annotate plans that should be considered for belief revision with `drp` (i.e. declarative rule plans).

```
@plan[drp] +!goal :  p & q & r <- +s.
```

When this plan is chosen, `s` will be added to the belief base. The revision algorithm will then be executed to revise any inconsistent pairs of literals.

The justification holds a support-list of type *Literal*, which in our case can be either *LiteralImpl* (beliefs) or *Plan* (declarative rule plans). Each justification has a quality – either assigned *a priori* (using a preference ordering as described above) or when propagating the quality.

Given the contraction and revision algorithms above, the Jason agent can now revise its belief base on-the-fly. Whenever a relevant plan is chosen which adds a belief A to the belief base, the **brf** method is executed[3], and it is checked whether an inconsistent pair $(B, \neg B)$ exists and should be revised.

The Jason platform also allows us to implement the revision algorithm directly in the agent architecture. We can then revise the belief base when a reasoning cycle is starting instead of revising during the addition or deletion of a belief, meaning that inconsistent information will never exist at the beginning of a reasoning cycle. Since we require belief-revised plans to be annotated with *DRP*, this solution could be relevant in a setting where it is difficult to decide exactly which plans could lead to inconsistency.

6 CASE STUDY — CONTINUED

We now consider the case study again. We choose the following preference ordering of independent beliefs and plans:

Percept
> Mental note
> Built-in plan
> Communicated data

[3]Note that in Jason the **brf** method is always executed when beliefs are added, but we ensure that the actual belief revision only occurs when the executed plan is in fact a DRP.

For simplicity we give each inferential justification a numerical value representing their quality corresponding to the above ordering: 0 for communicated data, 1 for built-in plans, 2 for mental notes and 3 for percepts.

The agent first receives `at(a)` $[Ag_2]$ and applies `+at(X) <- +dig(X)`, resulting in `dig(a)`. At this point the belief base is consistent. The justification for `dig(a)` is (`dig(a)`, [`at(a)`, `+at(X) <- +dig(X)`], 0), since the least preferred literal is `at(a)` ($p($`at(a)`$) = 0$).

It then receives ∼`at(a)` $[Ag_3]$, which is added to the belief base, but no plans apply. The agent now has the following information:

	Age	Belief
R1	0	`+at(X) <- +dig(X).`
B1	22	`at(a)` $[Ag_2]$
B2	22	`dig(a)` $[Ag_1]$
B3	28	∼`at(a)` $[Ag_3]$

This means that when **B3** is added to the belief base, the revision algorithm finds both `at(a)` $[Ag_2]$ and ∼`at(a)` $[Ag_3]$. The preference ordering is **B1** $\preceq$ **B3** even though $p($**B1**$) = p($**B3**$)$ since **B3** was added after **B1**. Therefore the agent contracts by `at(a)` $[Ag_2]$, resulting in the removal of `dig(a)` as well, since it depends on `at(a)`.

The agent perceives a plausible location, *b*, and adds `at(b)` $[Ag_1]$ to its belief base. By applying **R1** it also adds `dig(b)` $[Ag_1]$. It then receives ∼`at(b)` $[Ag_3]$ rendering the belief base inconsistent again:

	Age	Belief
R1	0	`+at(X) <- +dig(X).`
B3	28	∼`at(a)` $[Ag_3]$
B4	35	`at(b)` $[Ag_1]$
B5	35	`dig(b)` $[Ag_1]$
B6	44	∼`at(b)` $[Ag_3]$

We have **B6** $\preceq$ **B4** since $p($**B6**$) < p($**B4**$)$ so the agent will contract by ∼`at(b)` $[Ag_3]$.

The agent now receives a plan for retrieving a treasure located in a tree. It chooses to pursue this plan immediately, adding the following information:

	Age	Belief
R2	49	(`+!get_treasure : at(X) <- +climb_tree(X); +∼dig(X)`) $[Ag_4]$
B7	52	`climb_tree(b)` $[Ag_1]$
B8	52	∼`dig(b)` $[Ag_1]$

Figure 3 shows the relation between the current formulas in the agents belief base. We see that $p($**B8**$) < p($**B4**$)$, which means the agent will choose to contract by ∼`dig(b)` $[Ag_1]$. The agent will remove **B7**, **B8** and **R2** and then has the following information:

	Age	Belief
R1	0	`+at(X) <- +dig(X).`
B3	28	∼`at(a)` $[Ag_3]$
B4	35	`at(b)` $[Ag_1]$
B5	35	`dig(b)` $[Ag_1]$

The agents belief base is again consistent.

The example shows how a belief base can become inconsistent at any time and that it is relevant to consider plans as well when revising. The algorithms we have described can detect and remove inconsistency in the belief base, and it should be evident by the examples that the choice of preference ordering has great impact on the contents of the belief base after a revision has occurred.

7 CONCLUSIONS

We have successfully implemented revision of plans and beliefs in the AgentSpeak implementation Jason. The algorithm works in polynomial time, making it useful in practice. Furthermore we have given an example which shows the usefulness of revising both plans and beliefs in an agent's belief base.

We have built on the work presented in (Alechina et al., 2006a; Alechina et al., 2006b; Nguyen, 2009) by focusing on how plans can be added dynamically (either by being exchanged or derived) and has shown that this has practical use in Jason, because of its extensible nature.

We have shown that the choice of beliefs that should be removed has great influence on the result of a contraction, since reason-maintenance recurses through the dependency-graph.

An interesting extension to the contraction algorithm could be to allow non-inferential justifications to have a dynamic quality, meaning that percepts or built-in plans do not always have the same quality.

Furthermore, this paper has only looked at inconsistent pairs of the form $(A, \neg A)$, but it would be interesting to include means for discovering other types of inconsistency, such as $(dead(A), alive(A))$.

Other recent work on approaches to inconsistency handling in Jason and related systems should also be considered (Alechina et al., 2008; Klapiscak and Bordini, 2009; Villadsen, 2005; Fuzitaki et al., 2010; Mascardi et al., 2011; Jensen and Villadsen, 2012; Spurkeland et al., 2013).

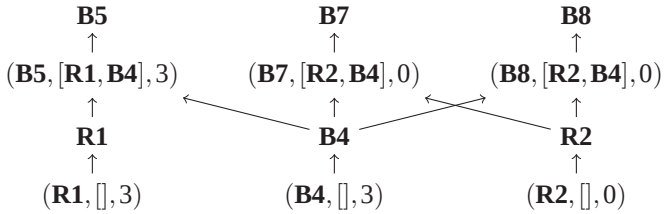


Figure 3: The relation between the agents beliefs after receiving a new plan for retrieving the treasure.

REFERENCES

Alchourron, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530.

Alechina, N., Bordini, R. H., Hübner, J. F., Jago, M., and Logan, B. (2006a). Automating belief revision for AgentSpeak. In *Proceedings of the 4th International Conference on Declarative Agent Languages and Technologies*, DALT'06, pages 61–77. Springer.

Alechina, N., Jago, M., and Logan, B. (2006b). Resource-bounded belief revision and contraction. In *Proceedings of the Third International Conference on Declarative Agent Languages and Technologies*, DALT'05, pages 141–154. Springer.

Alechina, N., Jago, M., and Logan, B. (2008). Preference-based belief revision for rule-based agents. *Synthese*, 165(1):159–177.

Bordini, R. H., Wooldridge, M., and Hübner, J. F. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons.

Doyle, J. (1977). Truth maintenance systems for problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, IJCAI 77, page 247.

Fuzitaki, C., Moreira, l., and Vieira, R. (2010). Ontology reasoning in agent-oriented programming. In da Rocha Costa, A., Vicari, R., and Tonidandel, F., editors, *Advances in Artificial Intelligence - SBIA 2010*, volume 6404 of *Lecture Notes in Computer Science*, pages 21–30. Springer.

Jensen, A. S. and Villadsen, J. (2012). Paraconsistent computational logic. In Blackburn, P., Jørgensen, K. F., Jones, N., and Palmgren, E., editors, *8th Scandinavian Logic Symposium*, pages 59–61. Scandinavian Logic Society.

Klapiscak, T. and Bordini, R. H. (2009). JASDL: A practical programming approach combining agent and semantic web technologies. In Baldoni, M., Son, T. C., Riemsdijk, M. B., and Winikoff, M., editors, *Declarative Agent Languages and Technologies VI*, pages 91–110. Springer.

Mascardi, V., Ancona, D., Bordini, R. H., and Ricci, A. (2011). CooL-AgentSpeak: Enhancing AgentSpeak-DL agents with plan exchange and ontology services. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2:109–116.

Nguyen, H. H. (2009). Belief revision in a fact-rule agent's belief base. In *Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, KES-AMSTA '09, pages 120–130. Springer.

Spurkeland, J., Jensen, A., and Villadsen, J. (2013). Belief revision in the GOAL agent programming language. *ISRN Artificial Intelligence*, 2013.

Villadsen, J. (2005). Supra-logic: Using transfinite type theory with type variables for paraconsistency. *Journal of Applied Non-Classical Logics*, 15(1):45–58.