Resource-bounded belief revision and contraction

Natasha Alechina, Mark Jago, and Brian Logan

School of Computer Science
University of Nottingham
Nottingham, UK
{nza,mtw,bsl}@cs.nott.ac.uk

Abstract. Agents need to be able to change their beliefs; in particular, they should be able to *contract* or remove a certain belief in order to restore consistency to their set of beliefs, and *revise* their beliefs by incorporating a new belief which may be inconsistent with their previous beliefs. An influential theory of belief change proposed by Alchourron, Gärdenfors and Makinson (AGM) [1] describes postulates which a rational belief revision and contraction operations should satisfy. The AGM postulates have been perceived as characterising idealised rational reasoners, and the corresponding belief change operations are considered unsuitable for implementable agents due to their high computational cost [3]. The main result of this paper is showing that an efficient (linear time) belief contraction operation nevertheless satisfies all but one of the AGM postulates for contraction. This contraction operation is defined for a realistic rule-based agent which can be seen as a reasoner in a very weak logic; although the agent's beliefs are deductively closed with respect to this logic, checking consistency and tracing dependencies between beliefs is not computationally expensive. Finally, we give a non-standard definition of belief revision in terms of contraction for our agent.

1 Introduction

Two main approaches to belief revision have been proposed in the literature: AGM (Alchourron, Gärdenfors and Makinson) style belief revision as characterised by the AGM postulates [1] and reason-maintenance style belief revision [2]. AGM style belief revision is based on the ideas of coherence and informational economy. It requires that the changes to the agent's belief state caused by a revision be as small as possible. In particular, if the agent has to give up a belief in A, it does not have to give up believing in things for which A was the sole justification, so long as they are consistent with the remaining beliefs. Classical AGM style belief revision describes an idealised reasoner, with a potentially infinite set of beliefs closed under logical consequence.

Reason-maintenance style belief revision, on the other hand, is concerned with tracking dependencies between beliefs. Each belief has a set of justifications, and the reasons for holding a belief can be traced back through these justifications to a set of foundational beliefs. When a belief must be given up, sufficient foundational beliefs have to be withdrawn to render the belief underivable. Moreover, if all the justifications for a belief are withdrawn, then that belief itself should no longer be held. Most implementations of reason-maintenance style belief revision are incomplete in the logical

M. Baldoni, U. Endriss, A. Omicini, and P. Torroni (eds.)
DALT 2005, Third International Workshop, Utrecht, July 25th, 2005
http://www.doc.ic.ac.uk/~ue/DALT-2005/

sense, but tractable. A more detailed comparison of the two approaches can be found in, for example, [3].

In this paper, we present an approach to belief revision and contraction for resource-bounded agents which is a synthesis of AGM and reason-maintenance style belief revision. We consider a simple agent consisting of a (finite) state and a (finite) agent program which executes in at most polynomial time. The agent's state contains literals representing the beliefs of the agent, and the agent's program consists of rules which are used to derive new beliefs from its existing beliefs. When the agent discovers an inconsistency in its beliefs, it removes sufficient beliefs (literals) to restore consistency. Our algorithm for belief contraction is similar to algorithms used for propagating dependencies in reason-maintenance systems, but we show that our approach satisfies all but one of the basic AGM postulates (the recovery postulate is not satisfied). The belief revision and contraction operations which we define compare in space and time complexity to the usual overhead of computing the conflict set and firing rules in a rule-based system. The basic contraction algorithm runs in time $O(kr + n)$, where n is the number of literals in the working memory, r is the number of rules and k is the maximal number of premises in a rule. We show how out algorithms can be adapted to remove the agent's least entrenched beliefs when restoring consistency. Recomputing entrenchment order of beliefs also has sub-quadratic complexity. Finally, we discuss definition of belief revision in terms of contraction for our agent, and show that using the Levi identity does not lead to the best result. We propose an alternative definition, and show that the resulting operation satisfies all but one of the basic AGM postulates for revision.

The paper is organised as follows. In Section 2, we introduce the AGM belief revision. In Section 3, we describe the rule-based resource-bounded reasoners. In Section 4, contraction algorithms for those reasoners is defined, and shown to run in linear time. The main result of the paper is in Section 5, where we define the logic under which the beliefs of our reasoners are closed, and show that the basic postulates for contraction, apart from recovery, hold for the contraction operations we defined. In Section 6, we show how to extend the algorithm to contract by a least preferred set of beliefs, using a preference order on the set of beliefs. In Section 8, we discuss related work, and in Section 9, we conclude.

2 AGM belief revision

The theory of belief revision as developed by Alchourron, Gärdenfors and Makinson in [4, 1, 5] models belief change of a idealised rational reasoner. The reasoner's beliefs are represented by a potentially infinite set of beliefs closed under logical consequence. When new information becomes available, the reasoner must modify its belief set to incorporate it. The AGM theory defines three operators on belief sets: expansion, contraction and revision. *Expansion*, denoted $K + A$, simply adds a new belief A to K and the resulting set is closed under logical consequence. *Contraction*, denoted by $K \dot{-} A$, removes a belief A from from the belief set and modifies K so that it no longer entails A. *Revision*, denoted $K \dot{+} A$, is the same as expansion if A is consistent with the cur-

rent belief set, otherwise it minimally modifies K to make it consistent with A, before adding A.

Contraction and revision cannot be defined uniquely, since in general there is no unique maximal set $K' \subset K$ which does not imply A. Instead, the set of 'rational' contraction and revision operators is characterised by the AGM postulates [1]. Below $Cn(K)$ denotes closure of K under logical consequence.

The basic AGM 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)

The basic postulates for revision are:

(K$\dot{+}$1) $K \dot{+} A = Cn(K \dot{+} A)$
(K$\dot{+}$2) $A \in K \dot{+} A$
(K$\dot{+}$3) $K \dot{+} A \subseteq K + A$
(K$\dot{+}$4) If $\{A\} \cup K$ is consistent, then $K + A = K \dot{+} A$[1]
(K$\dot{+}$5) $K \dot{+} A$ is inconsistent if, and only if, A is inconsistent.
(K$\dot{+}$6) If $Cn(A) = Cn(B)$, then $K \dot{+} A = K \dot{+} B$

The AGM theory elegantly characterises rational belief revision for an ideal reasoner. However it has been argued that the definition of the expansion, contraction and revision operators on belief sets and the resulting assumption of logical omniscience, means that it cannot be applied to resource-bounded reasoners. For example, Doyle [3] states: '...to obtain a practical approach to belief revision, we must give up both logical closure and the consistency and dependency requirements of the AGM approach' (p.42).

In the next section, we present a reasoner which has bounded memory and implements a polynomial (sub-quadratic) algorithm for belief contraction. In subsequent sections we show that it nevertheless satisfies the AGM postulates for rational belief contraction apart from ($K \dot{-} 5$). We achieve this by weakening the language and the logic of the reasoner. However, the reasoner is still realistic enough in that it corresponds to a typical rule-based agent.

3 Resource-bounded agents

We consider a simple resource-bounded agent consisting of a finite state and a finite agent program. The agent's state or working memory (WM) contains literals (propositional variables or their negations) representing the beliefs of the agent. The agent's program consists of a set of rules of the form:

$$A_1, \ldots, A_n \rightarrow B$$

[1] We replaced '$\neg A \notin K$' with '$\{A\} \cup K$ is consistent' here, since the two formulations are classically equivalent.

where $A_1, \ldots, A_n, B$ are literals. The agent repeatedly executes a sense-think-act cycle. At each cycle, the agent adds any observations (including communications from other agents) to its existing beliefs in WM and then fires its rules on the contents of WM.

We distinguish two models of rule application by the agent. In the simplest case, which we call the *quiescent setting* for belief revision, the agent fires all rules that match until no new rule instances can be generated (note that this takes at most polynomial time). In the quiescent setting, WM is closed under the agent's rules: all literals which can be obtained by the repeated application of rules to literals in the WM, are in WM. An example of a rule-based system which fires rules to quiescence is SOAR [6]. Another model of rule application, which is perhaps more interesting, is the non-quiescent case, which we call the *non-quiescent setting* for belief revision. In the non-quiescent setting, we look at revising the agent's beliefs after the application of one or more rules but before all the rule instances have been fired. This setting is natural when considering many rule-based systems, such as CLIPS [7], which fire one rule instance at a time.

Periodically, e.g., at the end of each cycle, or after each rule firing, the agent checks to see if its beliefs are consistent. If A is a literal, we denote by A^- the literal of the opposite sign, that is, if A is an atom p, then A^- is $\neg p$, and if A is a negated atom $\neg p$, then A^- is p. We say that WM is inconsistent iff for some literal A, both A and A^- are in WM. For each pair $\{A, A^-\} \subseteq WM$, the agent restores consistency by contracting by one element of each pair. Note that we only consider contraction by literals—rules are part of the agent's program and are not revised.

In the next section, we assume that the choice of whether to contract by A or A^- is arbitrary, as is the choice of literals to remove during contraction. Later we show how to incorporate preference orderings over the beliefs in WM to contract by a least preferred set of literals.

4 Contraction

We define resource-bounded contraction by a literal A as the removal of A and sufficient literals from WM so that A is no longer derivable using the rules which constitute the agent's program. In this section, we present a simple algorithm for resource-bounded contraction and show that it runs in time linear in $kr + n$, where r is the number of rules in the agent's program, k is the maximal number of premises in a rule, and n is the number of literals in the working memory.

We assume that WM consists of a list of cells. Each cell holds a literal and its associated dependency information in form of two lists, a dependencies list and a justifications list.[2] Both lists contain pointers to *justifications*, which correspond to fired rule instances; each justification records the derived literal, the premises of the rule, and (for efficiency's sake) back-pointers to the dependencies and justifications lists which reference them. We will denote a justification as $(A, [B\ C])$ or (A, s) where A is the derived literal and $[B\ C]$ or s is the list of premises of the rule (or *support* list). Each s has a distinguished position w which contains the 'weakest' member of s. Later we

[2] In the interests of brevity, we will refer to the cell containing the literal A as simply A when there is no possibility of confusion.

will show how to give a concrete interpretation to the notion of 'weakness' in terms of preferences; for now, we assume that w is the first position in s, or is chosen randomly.

The *dependencies list* of A contains the justifications for A. For example, the dependencies list $[(A, [B\ C])\ (A, [D])]$ means that A can be derived from B and C (together) and from D (on its own). In the quiescent setting, the dependencies list of A corresponds to all rules which have A in the consequent and where premises are actually in working memory. In the non-quiescent setting, the dependencies list corresponds only to the rules which have been fired so far. If A is an observation, or was present in WM when the agent started, its dependencies list contains a justification $(A, [\ ])$ with an empty support. The *justifications list* of A contains all the justifications where A is a member of a support. For example, if the dependencies list of C contains a justification $(C, [A\ B])$, then A's justifications list contains the justification $(C, [A\ B])$.

The dependencies and justifications lists are updated whenever a rule is fired. For example, when firing the rule $E, F \rightarrow B$, we check to see if B is in working memory, and, if not, add a new cell to WM containing the literal B. We also add the justification $(B, [E\ F])$ to the dependencies list for B and to the justifications lists of E and F.

The algorithm for contraction is very simple:

```
Algorithm: contraction by A

Loop 1: for each justification (C, s) in A's justifications list,
        remove (C, s) from C's dependencies list and from the
        justifications list of each literal in s.

Loop 2: for each justification (A, s) in A's dependencies list,
        if s is empty:
            remove (A, s);
        else:
            contract by the literal s[w];

Finally, delete the cell containing A.
```

The contraction algorithm consists of two main loops. Loop 1 traverses the justifications list, and for every justification in it, removes all references to the justification. We assume that removing a reference to a justification is a constant time operation, due to the use of back-pointers. If a justification corresponds to a rule with k premises, there are $k+1$ pointers to it: one from the dependencies list of the derived literal, and k from the justifications lists of the premises. Loop 2 traverses the dependencies list, and for each justification there, either removes it (if it has an empty support), or recurses to contract by the weakest member of the support list, w. The total number of steps the algorithm performs in the two loops is proportional to the total length of all dependencies and justifications lists involved. The maximal number of justification objects with non-empty supports is r, where r is the number of rules. The number of references to each justification object with a non-empty support is $k+1$, where k is the maximal number of premises in a rule. So the maximal number of steps is $r \times (k+1)$ for justifications with non-empty supports (assuming that each support can be updated in constant time), plus at most n for the justifications with empty supports, where n is the number of literals in WM. The last step in the contraction algorithm (removing a cell) is executed at

most n times, and we assume that access to a cell given its contents takes constant time. The total running time is therefore $O(kr + n)$.

4.1 Reason-maintenance type contraction

The algorithm above can be modified to perform reason-maintenance type contraction. Reason-maintenance contraction by A involves removing not just those justifications whose supports contain A, but also all beliefs which have no justifications whose supports do not contain A. In this case, in addition to removing the justifications in A's justifications list from other literals' dependencies lists, we check if this leaves the dependencies list of the justified literal empty. If so, we remove the justified literal and recurse forwards, following links in its justifications list. This adds another traversal of the dependencies graph, but the overall complexity remains $O(kr + n)$.

5 The agent's logic and AGM postulates

In this section, we present a weak logic W and show that our rule-based agent can be seen as a fully omniscient reasoner in W: namely, its belief set is closed with respect to derivability in W.

Consider a propositional language L_W where well-formed formulas are either (1) literals, or (2) formulas of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$, where $A_1, \ldots, A_n, B$ are literals. Note that there is a clear correspondence between an agent's rules and the second kind of formula. We will refer to the implication corresponding to a rule R as R, where it cannot cause confusion.

A logic W in the language L_W contains a single inference rule, generalised *modus ponens*:

$$\frac{A_1, \ldots, A_n, \quad A_1 \wedge \ldots \wedge A_n \rightarrow B}{B}$$

The notion of derivability in the logic is standard. We denote derivability in this logic by $\vdash_W$ and the corresponding consequence operator by C_W.

W is obviously much weaker than classical logic. In particular, the principle of excluded middle does not hold, so $A \rightarrow B$ and $A^- \rightarrow B$ do not imply B. For any finite set Γ of implications and literals, $C_W(\Gamma)$ is finite. It contains exactly the same implications and literals as Γ, plus possibly some additional literals derived by the inference rule. All such additional literals occur as consequents of the implications in Γ.

Let WM be the set of literals in working memory, and $\mathcal{R}$ the set of the agent's rules.

Proposition 1. *For any literal A, $WM \cup \mathcal{R} \vdash_W A$ iff $A \in WM$ after running $\mathcal{R}$ to quiescence.*

The proposition above means that the set comprising the agent's beliefs is closed under consequence if the agent runs all its rules to quiescence: $WM \cup \mathcal{R} = C_W(WM \cup \mathcal{R})$ after running $\mathcal{R}$ to quiescence.

Somewhat surprisingly, an agent which does not run its rules to quiescence can also be seen as a totally rational and omniscient reasoner in W — provided we only include the rules which actually have been fired in its beliefs.

Assume that a subset $\mathcal{R}'$ of the agent's rules $\mathcal{R}$ are fired.

Proposition 2. *Let $\mathcal{R}' \subseteq \mathcal{R}$; then for any literal A, $WM \cup \mathcal{R}' \vdash_W A$ iff $A \in WM$ after firing the rules $\mathcal{R}'$.*

In other words, in the non-quiescent setting, $WM \cup \mathcal{R}' = C_W(WM \cup \mathcal{R}')$ where $\mathcal{R}'$ is the set of rules fired.

By the belief state K of the agent we will mean the set of literals in its working memory and the set of rules which have been fired.

$$K \stackrel{df}{=} C_W(WM \cup \mathcal{R}') \tag{1}$$

By $K \dot{-} A$ we will denote the result of applying our contraction by A algorithm to K.

Now we can show that AGM belief postulates are satisfied for our agent.

Proposition 3. *$\dot{-}$ satisfies (K$\dot{-}$1)–(K$\dot{-}$4) and (K$\dot{-}$6).*

Proof: Given that K is closed under consequence, and contraction removes literals and recursively destroys rule instances used to derive them, no new rule instances can be generated after the algorithm completes. So $K \dot{-} A$ is still closed under consequence and K$\dot{-}$1 holds. K$\dot{-}$2 holds because $\dot{-}$ deletes literals from the working memory without adding any, K$\dot{-}$3 is satisfied for the same reason. K$\dot{-}$4 states that after a contraction by A, A is no longer in the working memory. Since the contraction algorithm removes A as its last step, A is indeed not in the working memory after the algorithm is executed. K$\dot{-}$6 is trivially valid, since for any literal A, $C_W(A) = \{A\}$. □

Proposition 4. *$\dot{-}$ does not satisfy K$\dot{-}$5.*

Proof: Suppose we have a single rule $R = A \rightarrow B$ and $WM = \{A, B\}$. After contraction by B, WM is empty. When we expand by B, WM contains only B. □

The recovery postulate is the most controversial of the AGM postulates [8], and many contraction operations defined in the literature do not satisfy it. We can satisfy K$\dot{-}$5 in our setting, if we are prepared to re-define the expansion operator. We simply save the current state of working memory before a contraction, and restore the previous state of WM if we have a contraction followed by an expansion by the same literal. More precisely, to expand by a literal A, we first check if the previous operation was contraction by A, and if so we restore the previous state of working memory. Otherwise we add A to the contents of WM and run (a subset of) the agent's rules. This requires $O(n)$ additional space, where n is the size of the working memory.

6 Preferred contractions

So far we have assumed that the choice of literals to be removed during contraction is arbitrary. However, in general, an agent will prefer some contractions to others. For

example, an agent may prefer to remove the smallest set of beliefs necessary to restore consistency, or to remove those beliefs which are least strongly believed. The problem of computing a minimal set of beliefs which, if deleted, would restore consistency is exponential in the size of working memory, and approaches based on this type of 'minimal' contraction and revision do not sit comfortably within our resource-bounded framework. In this section we focus instead on contractions based on preference orders over individual beliefs, e.g., degree of belief or commitment to beliefs. We show that computing the most preferred contraction can be performed in time linear in $kr + n$.

We distinguish *independent* beliefs, beliefs which have at least one non-inferential justification (i.e., a justification with an empty support), such as observations and the literals in working memory when the agent starts. We assume that an agent associates an *a priori* quality with each non-inferential justification for its independent beliefs. For example, communicated information may be assigned a degree of reliability by its recipient which depends on the degree of reliability of the speaker; percepts may be assumed to be more reliable than communicated information and so on. For simplicity, we assume that 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. A value of 0 means lowest quality and m means highest quality. We take the preference of a literal A, $p(A)$, to be that of its highest quality justification:

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

where $j_0, \ldots, j_n$ are all the justifications for A, and define the quality of an inferential justification to be that of the least preferred belief in its support:[3]

$$qual(j) = min\{p(A) : A \in \text{ support of } j\}.$$

This is similar to ideas in argumentation theory: an argument is only as good as its weakest link, yet a conclusion is at least as good as the best argument for it. This approach is also related to Williams 'partial entrenchment ranking' [9] which assumes that the entrenchment of any sentence is the maximal quality of a set of sentences implying it, where the quality of a set is equal to the minimal entrenchment of its members. While this approach is intuitively appealing, nothing hangs on it, and any preference order over literals is consistent with the postulates. For example, the preference of a derived literal could be a property of the rule or given by some function of the preferences of its antecedents.

To perform a preferred contraction, we preface the contraction algorithm given above with a step which computes the preference of each literal in WM and for each justification, finds the position of a least preferred member of support. We conduct this process in stages, starting with the most preferred independent beliefs. Note that unless WM is empty, it always contains at least one literal with a justification whose support is empty (otherwise nothing could be used to derive other literals) and at least one of those independent literals is going to have the maximal preference value of literals in

[3] For simplicity, in what follows we assume reason-maintenance style contraction. To compute preferences for coherence-style contraction we can assume that literals with no supports (as opposed to an empty support) are viewed as having an empty support of lowest quality.

WM even when all other preferences are computed (since a derived literal cannot have a higher preference than all of the literals in justifications for it). Assume we have a list `ind` of justifications with an empty support `(A,[],q)`, where `q` is the quality of the empty support. We associate a counter `c(j)` with every justification `j = (A, s)`. Initially `c(j)` is set to be the length of `s`. When `c(j)` is 0, the preferences of all literals in `s` have been set.

```
Algorithm: preference computation

Order ind in descending order by q.

While there is j=(A,[],q) in ind with A unmarked, repeat:

   take first unmarked j=(A,[],q) in ind
   mark A; p(A) = q; propagate(A,q)

Procedure: propagate(A, q)

for each j =(B,s) in A's justifications list, decrement c(j);

if c(j) = 0:
  qual(j) = q;
  w(s) = A's position in s;

  if B is unmarked:
     mark B,
     p(B) = q;
     propagate(B,q)
```

We then simply run the contraction algorithm, to recursively delete the weakest member of each support in the dependencies graph of A.

We define the *worth* of a set of literals as $worth(\Gamma) = max\{p(A) : A \in \Gamma\}$. We can prove that our contraction algorithm removes the set of literals with the least worth.

Proposition 5. *If WM was contracted by A and this resulted in removal of the set of literals Γ, then for any other set of literals Γ' such that $WM - \Gamma'$ does not imply A, $worth(\Gamma) \leq worth(\Gamma')$.*

Proof: if $A \notin WM$, the statement is immediate since $\Gamma = \emptyset$. Assume that $A \in WM$. In this case, $A \in \Gamma$ and $A \in \Gamma'$ (otherwise $WM - \Gamma$ and $WM - \Gamma'$ would still derive A). It is also easy to see that A is the maximal element of Γ, because a literal B is in Γ if either (1) $B = qual(j_i)$ for some justification j_i for A, and since $p(A) = max(qual(j_0), ..., qual(j_n))$, $p(B) \leq p(A)$; or (2) B is a least preferred element of a support set for some literal A depends on, in which case its preference is less or equal to the preference of the literal it is justification for, which in turn is less or equal to $p(A)$. So, since A is an element of both Γ and Γ', and A has the maximal preference in Γ, then $worth(\Gamma) \leq worth(\Gamma')$. □

Computing preferred contractions involves only modest computational overhead. The ordering of `ind` takes $O(n \log n)$ steps; `ind` is traversed once, which is $O(n)$;

`propagate` traverses each justifications list once, which is $O(kr)$ (setting the w index in each support can be done in constant time, assuming that the justifications list of each literal A actually contains pairs, consisting of a justification and the index of A's position in the support list of the justification). The total cost of computing the preference of all literals in WM is therefore $O(n \log n + kr)$. As the contraction algorithm is unchanged, this is also the additional cost of computing a preferred contraction.

7 Revision

In the previous sections we described how to contract by a belief. Now let us consider revision, which is adding a new belief in a manner which does not result in an inconsistent set of beliefs. For simplicity, we will consider revision in a quiescent setting only.

If the agent is a reasoner in classical logic, revision is definable in terms of contraction and vice versa. Given a contraction operator $\dot{-}$ which satisfies postulates (K$\dot{-}$1)–(K$\dot{-}$4) and (K$\dot{-}$6), a revision operator $\dot{+}$ defined as $K \dot{+} A \stackrel{df}{=} (K \dot{-} \neg A) + A$ (Levi identity) satisfies postulates (K$\dot{+}$1)–(K$\dot{+}$6). Conversely, if a revision operator satisfies (K$\dot{+}$1)–(K$\dot{+}$6), then contraction defined as $K \dot{-} A \stackrel{df}{=} (K \dot{+} \neg A) \cap K$ (Harper identity) satisfies postulates (K$\dot{-}$1)–(K$\dot{-}$6) (see [5]).

However, revision and contraction are not inter-definable in this way for an agent which is not a classical reasoner, in particular, a reasoner in a logic for which it does not hold that $K + A$ is consistent if, and only if, $K \not\vdash A^-$. If we apply the Levi identity to the contraction operation defined earlier, we will get a revision operation which does not satisfy the revision postulates. One of the reasons for this is that contracting the agent's belief set by A^- does not make this set consistent with A, so $(K \dot{-} A^-) + A$ may be inconsistent.

Let us instead define revision by A as $(K+A) \dot{-} \bot$ (add A, close under consequence and eliminate all contradictions)

```
Algorithm: revision by A

Add A to WM;
Run rules to quiescence;

while there is a pair (B, B-) in WM:
   contract by the least preferred member of the pair
```

However, even for this definition of revision, not all basic AGM postulates are satisfied.

Proposition 6. *The revision operation defined above satisfies (K$\dot{+}$1) and (K$\dot{+}$3) – (K$\dot{+}$6).*

Proof. (K$\dot{+}$1) is satisfied because when we do $\dot{+}$, we run the rules to quiescence. (K$\dot{+}$3) is satisfied because the construction of $K \dot{+} A$ starts with A being added to WM which is then closed under consequence (which is $K + A$), and after that literals can only be removed from WM. (K$\dot{+}$4) holds because, if adding A does not cause an inconsistency,

then $K \dot{+} A = K + A$ by the definition of $\dot{+}$. (K$\dot{+}$5) holds trivially because A and $K \dot{+} A$ are never inconsistent. Finally, recall that in the agent's logic, $Cn(A) = Cn(B)$ only if $A = B$, so (K$\dot{+}$6) holds trivially.

The reason why (K$\dot{+}$2), or the property that $A \in K \dot{+} A$, does not hold, is simple. Suppose we add A to K and derive some literal B, but B^- is already in WM and has a high preference value (higher than B). Then we contract by B, which may well result in contraction by A.

One could question whether (K$\dot{+}$2) is a desirable property; it has been argued in [10] that an agent which performs autonomous belief revision would not satisfy this postulate in any case. However, if we do want to define a belief revision operation which satisfies this postulate, we need to make sure that A has a higher preference value than anything else in working memory, and that A on its own cannot be responsible for an inconsistency. One way to satisfy the first requirement is to use a preference order based on timestamps: more recent information is more preferred. To satisfy the second requirement, we may postulate that the agent's rules are not *perverse*. We call a set of rules $\mathcal{R}$ perverse if there is a literal A such that running $\mathcal{R}$ to quiescence on $WM = \{A\}$ results in deriving a contradiction $\{B, B^-\}$ (including the possibility of deriving A^-). This is equivalent to saying that no singleton set of literals is *exceptional* in the sense of [11].

8 Related work

AGM belief revision is generally considered to apply only to idealised agents, because of the assumption that the set of beliefs is closed under logical consequence. To model Artificial Intelligence agents, an approach called belief base revision has been proposed (see for example [12–15]). A belief base is a finite representation of a belief set. Revision and contraction operations can be defined on belief bases instead of on logically closed belief sets. However the complexity of these operations ranges from NP-complete (full meet revision) to low in the polynomial hierarchy (computable using a polynomial number of calls to an NP oracle which checks satisfiability of a set of formulas) [16]. This complexity would not generally be considered appropriate for operations implemented by a resource-bounded agent. The reason for the high complexity is the need to check for classical consistency while performing the operations. One way around this is to weaken the language and the logic of the agent so that the consistency check is no longer an expensive operation (as suggested in [17]). This is the approach taken in this paper.

Our contraction algorithm is similar to the algorithm proposed by McAllester in [18] for boolean constraint propagation. McAllester also uses a notion of 'certainty' of a node, which is similar to our definition of preference.

Our approach to defining the preference order on beliefs is similar to the approach developed in [19, 20, 9] by Williams, Dixon and Wobcke. However, since they work with full classical logic, and calculating entrenchment of a sentence involves considering all possible derivations of this sentence, the complexity of their contraction and revision operations is at least exponential.

The motivation of our work is very similar to Wasserman's in [21], but Wasserman's solution to the computational cost of classical belief revision is to consider only small (relevant) subsets of belief base and do classical belief revision on them. Chopra et al [22], defined a contraction operation which approximates a classical AGM contraction operation; its complexity is $O(|K| \cdot |A| \cdot 2^S)$, where K is the knowledge base, A the formula to be contracted, and S is a set of 'relevant' atoms. As S gets larger, the contraction operation becomes a closer approximation of the classical contraction.

Perhaps the work most similar to our is that of Bezzazi et al [11], where belief revision and update operators for forward chaining reasoners were defined and analysed from the point of view of satisfying rationality postulates. The operators are applied to programs, which are finite sets of rules and literals, and are presented as 'syntactic' operators, which do not satisfy the closure under consequence and equivalence postulates. Rather, the authors were interested in preserving the 'minimal change' spirit of revision operators, which resulted in algorithms with high (exponential) complexity. Only one of the operators they proposed, *ranked revision*, has polynomial complexity. To perform a ranked revision of a program P by a program P', a base $\langle P_0, \ldots, P_n = \emptyset \rangle$ of P is computed, where $P_0 = P$ and each P_{i+1} is a subset of P_i containing the 'exceptional' rules of P_i. The base of a program can be computed in polynomial time in the size of the program; this involves adding the premises of each rule to the program, running rules to quiescence and checking for consistency. If the resulting set is inconsistent, the rule is exceptional. Ranked revision of P by P' is defined as $P_i \cup P'$, where P_i is the largest member of the base of P which is consistent with P. Consistency can also be checked in polynomial time by running the program to quiescence. For programs without exceptional rules, the result of ranked revision is either the union of P and P', if they are consistent, or P' alone, which is essentially full meet contraction. Even for the full classical logic, this is computable in NP time.

9 Conclusions and further work

In this paper, we have presented a realisable resource-bounded agent which does AGM style belief revision. The agent is rule-based, and can be seen as a fully rational and omniscient reasoner in a very weak logic. The rules of the agent's program are fixed, and only literal beliefs are revised. We define an efficient algorithm for contraction, similar to McAllester's algorithm for boolean constraint propagation, and show that the corresponding contraction operation satisfies all the basic AGM postulates apart from the recovery postulate. We show how to use a preference order on beliefs similar to entrenchment ranking introduced in [9] to contract with the minimally preferred set of beliefs. The additional cost of computing the preference order is small: the resulting algorithm is still sub-quadratic in the size of the agent's program. We then define a belief revision operator in terms of contraction, and show that it also satisfies all but one of the basic AGM postulates. The complexity of belief revision is polynomial in the size of the agent's program and the number of literals in the working memory. To the best of our knowledge, no one has previously pointed out that reason-maintenance style belief revision satisfies the AGM rationality postulates, provided that we assume that the logic which the agent uses is weaker than full classical logic.

In future work, we plan to look at efficient revision operations on the agent's programs, and extend the syntax of the agent's programs.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet functions for contraction and revision. Journal of Symbolic Logic **50** (1985) 510–530
2. Doyle, J.: Truth maintenance systems for problem solving. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence, IJCAI 77. (1977) 247
3. Doyle, J.: Reason maintenance and belief revision. Foundations vs coherence theories. In Gärdenfors, P., ed.: Belief Revision. Volume 29. Cambridge University Press, Cambridge, UK (1992) 29–51
4. Gärdenfors, P.: Conditionals and changes of belief. In Niiniluoto, I., Tuomela, R., eds.: The Logic and Epistemology of Scientific Change. North Holland (1978) 381–404
5. Gärdenfors, P.: Knowledge in Flux: Modelling the Dynamics of Epistemic States. The MIT Press, Cambridge, Mass. (1988)
6. Laird, J.E., Newell, A., Rosenbloom, P.S.: SOAR: An architecture for general intelligence. Artificial Intelligence **33** (1987) 1–64
7. Software Technology Branch, Lyndon B. Johnson Space Center Houston: CLIPS Reference Manual: Version 6.21. (2003)
8. Makinson, D.: On the status of the postulate of recovery in the logic of theory change. Journal of Philosophical Logic **16** (1987) 383–394
9. Williams, M.A.: Iterated theory base change: A computational model. In: Proceedings of Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), San Mateo, Morgan Kaufmann (1995) 1541–1549
10. Galliers, J.R.: Autonomous belief revision and communication. In Gärdenfors, P., ed.: Belief Revision. Cambridge Tracts in Theoretical Computer Science 29. Cambridge University Press (1992) 220–246
11. Bezzazi, H., Janot, S., Konieczny, S., Pérez, R.P.: Analysing rational properties of change operators based on forward chaining. In Freitag, B., Decker, H., Kifer, M., Voronkov, A., eds.: Transactions and Change in Logic Databases. Volume 1472 of Lecture Notes in Computer Science., Springer (1998) 317–339
12. Makinson, D.: How to give it up: A survey of some formal aspects of the logic of theory change. Synthese **62** (1985) 347–363
13. Nebel, B.: A knowledge level analysis of belief revision. In Brachman, R., Levesque, H.J., Reiter, R., eds.: Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference, San Mateo, Morgan Kaufmann (1989) 301–311
14. Williams, M.A.: Two operators for theory base change. In: Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence, World Scientific (1992) 259–265
15. Rott, H.: "Just Because": Taking belief bases seriously. In Buss, S.R., Hájaek, P., Pudlák, P., eds.: Logic Colloquium '98—Proceedings of the 1998 ASL European Summer Meeting. Volume 13 of Lecture Notes in Logic., Association for Symbolic Logic (1998) 387–408
16. Nebel, B.: Base revision operations and schemes: Representation, semantics and complexity. In Cohn, A.G., ed.: Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI'94), Amsterdam, The Netherlands, John Wiley and Sons (1994) 341–345
17. Nebel, B.: Syntax-based approaches to belief revision. In Gärdenfors, P., ed.: Belief Revision. Volume 29. Cambridge University Press, Cambridge, UK (1992) 52–88
18. McAllester, D.A.: Truth maintenance. In: Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90), AAAI Press (1990) 1109–1116

19. Dixon, S.: A finite base belief revision system. In: Proceedings of Sixteenth Australian Computer Science Conference (ACSC-16): Australian Computer Science Communications. Volume 15., Brisbane, Australia, Queensland University of Technology, Australia (1993) 445–451
20. Dixon, S., Wobcke, W.: The implementation of a first-order logic AGM belief revision system. In: Proceedings of 5th IEEE International Conference on Tools with AI, Boston, MA, IEEE Computer Society Press (1993) 40–47
21. Wasserman, R.: Resource-Bounded Belief Revision. PhD thesis, ILLC, University of Amsterdam (2001)
22. Chopra, S., Parikh, R., Wassermann, R.: Approximate belief revision. Logic Journal of the IGPL **9** (2001) 755–768