# The Implementation of a First-Order Logic AGM Belief Revision System

Simon Dixon and Wayne Wobcke
Knowledge Systems Group
Basser Department of Computer Science
University of Sydney, NSW 2006, Australia
e-mail: simon@cs.su.oz.au, wobcke@cs.su.oz.au

**Abstract**

*Belief revision is increasingly being seen as central to a number of fundamental problems in Artificial Intelligence such as nonmonotonic reasoning, reasoning about action, truth maintenance and database update. This paper describes the first implementation of an AGM belief revision system. The system is based on classical first-order logic, and for any finitely representable belief state, it efficiently computes expansions, contractions and revisions satisfying the AGM postulates for rational belief change. The system uses a finite base to represent a belief set, and interprets a partially specified entrenchment as representing a unique 'most conservative' entrenchment – this is motivated by considerations of evidence and by the close connections between belief revision and nonmonotonic reasoning. We describe in detail the algorithms for belief change, and give some examples of the system's operation.*

## 1. Introduction

Belief revision is the process of an agent's modifying its beliefs in the light of new information. If the information conflicts with the agent's current beliefs, the agent must give up some existing beliefs in order to accept it: in this case, the agent must make the appropriate choice of which beliefs to give up. The problem of belief revision has received considerable attention not least because of its close relation to the semantics of indicative conditionals and to scientific theory formation.

In Artificial Intelligence, belief revision is increasingly being seen as central to a number of fundamental problems. Belief revision has been shown to underlie nonmonotonic reasoning [8]; this connection is strengthened in [16,17] which relate belief revision to conditional logic approaches to nonmonotonic reasoning. Gärdenfors [9] shows the connections with the default reasoning of Poole [13] and belief revision has been shown to be formally as powerful as truth maintenance systems in [5]. Ideas from belief revision have also been used in [11] to give a semantics for events; this was applied to reasoning about action in [12] to give an account of the frame, ramification and qualification problems. Belief revision is also naturally suited to modelling updates of logical databases.

Current approaches to formalizing belief revision are based on mathematical methods [1], in which a belief state is modelled as a set of logical formulae and a belief revision operation is a function, from belief sets and formulae to belief sets, that satisfies a number of 'rationality postulates'. Known as the AGM approach (named after the authors of [1]), this method is non-computational in the sense that there are many belief revision functions satisfying the postulates, but no way of specifying a particular intended belief revision function was given. This problem has since been addressed by [7] which showed how a pre-order on formulae, called an epistemic entrenchment, could be used to specify which beliefs to give up on the acquisition of new, conflicting, information. The approach is still non-computational to the extent that a pre-order on all formulae in the language cannot necessarily be represented directly on a computer.

Although the formal properties of AGM belief revision functions have been much studied, there have been few attempts to specify computational versions of belief revision systems. Nebel [10] presents an approach based on epistemic relevance, which is closely related to epistemic entrenchment, but his revision functions are syntax-dependent and thus do not satisfy all the AGM postulates. Williams [15] shows how to define revision functions using a finite ensconcement of formulae, which is also similar to an entrenchment, although again, her contraction functions do not satisfy the AGM postulates. [4] describes an implementation of Williams' belief revision operations based on ensconcement.

In this paper we present a computational model of the AGM approach to belief revision and describe its implementation over the language of first-order logic. The two theoretical problems addressed are (i) the specification of the contents of a belief state: here a finite base together with a sound and complete inference procedure is used to represent a belief set, and (ii) the specification of a particular revision function: our system uses a finite partially specified entrenchment to represent belief change operations. A fundamental problem is how to extend a finite portion of an entrenchment to an entrenchment so that a belief revision function is properly specified (an entrenchment is a pre-order over all formulae in the logical language). We propose that a partially specified entrenchment stands for a canonical entrenchment called in [16] the most conservative entrenchment. Most conservative entrenchments are motivated both by considerations of entrenchments as representing evidence for beliefs and by connections with nonmonotonic reasoning. In addition, we specify efficient algorithms for the implementation of belief revision operations and describe some details of the belief revision system we have implemented.

The paper is organized as follows. In section 2, we summarize the AGM belief revision logic. In section 3, we present our computational version of the AGM approach, which forms the basis of our algorithms for belief revision, described in section 4. Some examples of the system's operation are presented in section 5. We conclude the paper with a discussion of what has been achieved and describe the direction of our future research.

## 2. The AGM approach to belief revision

In the AGM approach to belief revision [1], belief states are modelled using a logical language which is constrained only to be compact, consistent, contain the Propositional Calculus and modus ponens, and satisfy the deduction theorem. A belief set is a logically closed set of formulae over this logic. There are three operations specified on belief sets: expansion, contraction and revision, each of which must satisfy a number of 'rationality postulates'.

### 2.1 Three operations for belief change

The *expansion* of the belief set $K$ by a formula $\alpha$, denoted $K^+_\alpha$, represents the addition of $\alpha$ to $K$ regardless of whether it is consistent with $K$. Writing $Cn$ for the logical consequence operator, the postulates for expansion imply that $K^+_\alpha = Cn(K \cup \{\alpha\})$.

The *contraction* of a belief set with respect to a formula $\alpha$, denoted $K^-_\alpha$, is intended to model the removal of $\alpha$ from a belief set so that it is no longer a consequence of that set. Thus in removing $\alpha$, any other formulae which imply $\alpha$ must also be removed from the belief set. Contraction functions must satisfy the following postulates:

(K-1) $K^-_\alpha = Cn(K^-_\alpha)$
(K-2) $K^-_\alpha \subseteq K$
(K-3) If $\alpha \notin K$ then $K^-_\alpha = K$
(K-4) If $\not\vdash \alpha$ then $\alpha \notin K^-_\alpha$
(K-5) $K \subseteq (K^-_\alpha)^+_\alpha$
(K-6) If $\vdash \alpha \leftrightarrow \beta$ then $K^-_\alpha = K^-_\beta$
(K-7) $K^-_\alpha \cap K^-_\beta \subseteq K^-_{\alpha\wedge\beta}$
(K-8) If $\alpha \notin K^-_{\alpha\wedge\beta}$ then $K^-_{\alpha\wedge\beta} \subseteq K^-_\alpha$

The *revision* of a belief set $K$ by a formula $\alpha$, denoted $K^*_\alpha$, models the change made to $K$ in order to accept new information $\alpha$. Revision functions must satisfy the following postulates:

(K*1) $K^*_\alpha = Cn(K^*_\alpha)$
(K*2) $\alpha \in K^*_\alpha$
(K*3) $K^*_\alpha \subseteq K^+_\alpha$
(K*4) If $\neg\alpha \notin K$ then $K^+_\alpha \subseteq K^*_\alpha$
(K*5) $K^*_\alpha = K_\perp$ only if $\vdash \neg\alpha$
(K*6) If $\vdash \alpha \leftrightarrow \beta$ then $K^*_\alpha = K^*_\beta$
(K*7) $K^*_{\alpha\wedge\beta} \subseteq (K^*_\alpha)^+_\beta$
(K*8) If $\neg\beta \notin K^*_\alpha$ then $(K^*_\alpha)^+_\beta \subseteq K^*_{\alpha\wedge\beta}$

The postulates for contraction and revision do not uniquely determine contraction and revision functions. However, a revision can be seen as a two stage process, first a contraction of $K$ by the formula $\neg\alpha$ to remove beliefs inconsistent with $\alpha$, then an expansion by $\alpha$. The correctness of this process is guaranteed by the Levi identity: $K^*_\alpha = (K^-_{\neg\alpha})^+_\alpha$. Similarly, a contraction function is definable from a revision function by the Harper identity: $K^-_\alpha = K \cap K^*_{\neg\alpha}$.

### 2.2 Epistemic entrenchment

Given a collection of belief sets over some logic, there are many possible contraction and revision functions satisfying the AGM postulates. To specify particular revision and contraction functions, [7] uses a pre-order on all formulae of the logic, called an *epistemic entrenchment*. The conditions for an ordering $\leq$ to be an entrenchment are:

(EE1) If $A \leq B$ and $B \leq C$ then $A \leq C$
(EE2) If $A \vdash B$ then $A \leq B$
(EE3) For any A and B, $A \leq A\wedge B$ or $B \leq A\wedge B$
(EE4) When $K \neq K_\perp$, $A \notin K$ iff $A \leq B$ for all B
(EE5) If $B \leq A$ for all B then $\vdash A$

An entrenchment on a language specifies a unique contraction function defined using the (C–) condition:

$\beta \in K^-_\alpha$ iff $\beta \in K$ and either $\vdash \alpha$ or $\alpha < \alpha \vee \beta$

The revision function can then be defined uniquely from the Levi identity, or directly, using the (C*) condition:

$\beta \in K^*_\alpha$ iff either $\vdash \neg\alpha$ or $\neg\alpha < \alpha \rightarrow \beta$

## 3. A computational approach to AGM belief revision

In order to implement an AGM belief revision system, a representation for both belief sets and belief change operations must be found. The obvious choice for representing the belief set is that of a finite set of formulae, a finite *base*, from which all other beliefs can be derived using a sound and complete decision procedure for logical consequence. Of course for first-order logic, completeness of the decision procedure for derivability must be given up: we describe how our system handles this in section 4.

Particular contraction and revision functions can be specified using an epistemic entrenchment, but it is not possible to represent all entrenchments directly: some entrenchments allow infinitely many degrees of strength of beliefs. Moreover, it is impossible for the user of a system to enter all entrenchment relations: a more compact representation must be found. We choose to specify belief revision functions using a finite partially specified entrenchment, which is taken by our system to extend in a definite way to a uniquely specified *most conservative* entrenchment.

### 3.1 Most conservative entrenchments

The idea behind most conservative entrenchments is the following. The relative entrenchment of a formula with respect to other formulae is intended to represent the degree of evidence possessed by the system for a particular belief. A representation of a belief state is therefore a collection of formulae (those of the base) together with the degrees of evidence for each formula in the base. The belief set represented is the logical closure of the belief set base. Similarly, the most conservative entrenchment is the ‘closure’, in some sense, of the entrenchment ‘base’. To define this sense of closure, we take it that the only evidence possessed by the system for a belief is that derived from the evidence for formulae in the base: the evidence for a formula derived from a non-redundant collection of formulae is the evidence for the conjunction of these formulae, which is just the evidence for the weakest formula in the conjunction. In this sense, the generated entrenchment is conservative in not attributing evidence for a belief other than that warranted by the evidence for the base beliefs from which that belief derives.

The most conservative entrenchment generated from an entrenchment base can be formalized as follows. For convenience, we represent the degree of evidence for a belief as a natural number known as the *rank* of the formula, so a belief set is a set of formulae each with an associated rank. The higher the rank, the more evidence there is for a belief, and logical theorems have the highest rank of all. The entrenchment base $\Gamma$ can then be represented by a partition of the formulae in the base, called the ensconcement ordering in [15], determined by the subsets of formulae of the base all of which have the same rank. Suppose the logical theorems have rank $n$, and let the partitions be $\Gamma_1, \Gamma_2, \cdots, \Gamma_n$, some of which may be empty. The rank of a formula in the base is just the index of the partition to which it belongs. To define the most conservative entrenchment generated from this partition, we extend the definition of rank to all formulae. First let $\overline{\Gamma_i} = \bigcup_{j=i}^{n} \Gamma_j$. Intuitively, $\overline{\Gamma_i}$ is the set of formulae in the base for which there is degree of evidence at least $i$. Then the rank of a belief $A$, representing the degree of evidence for $A$, is the largest $i$ such that $A$ is a consequence of $\overline{\Gamma_i}$:

$$rank(A) = max(\{i \mid \overline{\Gamma_i} \vdash A\}) \text{ if } A \in K$$

The rank of a non-belief is defined to be 0.

The most conservative entrenchment determined by the partition of the base is then given by:

$$A \leq B \quad \text{iff} \quad rank(A) \leq rank(B)$$

To ensure consistency with the postulates for entrenchments (EE1–EE5), the following conditions on the entrenchment base must hold:

(R1) $\forall B \in \Gamma, \{A \in \Gamma \mid rank(B) < rank(A)\} \nvdash B$

(R2) $rank(A) = n \quad \text{iff} \quad \vdash A$

It can be shown that condition (R1) implies entrenchment axioms (EE2) and (EE3), and condition (R2) implies (EE5). (EE1) comes directly as a result of representing the rank by integers, and (EE4) follows from the definition of rank. Thus the most conservative entrenchment is an epistemic entrenchment.

We note two representation theorems of [15]:

1. A belief set can be represented by a finite base if and only if it is finitely axiomatizable.
2. An entrenchment can be represented by an entrenchment base if and only if it has a finite number of natural partitions each of which is finitely axiomatizable.

If an epistemic state fulfills both of these conditions, we say it is *finitely representable*. Our system can be used to implement any finitely representable AGM belief revision system based on first-order logic. (Note that this result relies on the choice of entrenchment revision policy. Using the minimal change policy adopted in this paper, the application of any AGM operator to a finitely representable belief state always yields another finitely representable belief state.)

### 3.2 Entrenchment revision

A major weakness of the AGM approach to belief revision is that the operations are defined as mapping belief sets and formulae to belief sets. Although these operations can be specified using epistemic entrenchments, there are no constraints placed on the entrenchments of formulae in the belief sets resulting from revisions, contractions or expansions. This means that it is not possible to perform two successive contraction or revision operations unless the entire entrenchment is respecified. It is possible to form the composition of a revision or contraction with an expansion (which is independent of the entrenchment).

Spohn [14] was the first to address this problem, within a more general framework than the AGM logic. In his work, an *epistemic state* is not represented directly by a set of formulae with an associated entrenchment, but indirectly, by an Ordinal Conditional Function (OCF), which gives an ordering on the models of the language by assigning an ordinal number to each model. Spohn uses only one operation to perform all changes to epistemic states – expansion, contraction, revision and change of entrenchment. The result of such an operation is another OCF, so that successive revisions and contractions may be performed. Another feature of this work is that all epistemic changes are reversible. Thus Spohn defines a model-theoretic approach to belief revision, but there is no known syntactic analogue of this which could be used for implementation.

We therefore adopt an alternative approach to specifying the entrenchment of a belief set resulting from a belief change operation. Our idea is to interpret the principle of minimal change, formulated in [6] as applying to the contents of a belief set, as also applying to the entrenchment of formulae in the belief set. This is justified by interpreting the rank of a belief in our system as representing the degree of evidence for that belief. Under this interpretation, if a belief is not affected by a belief change operation, its degree of evidence should not be affected either. That is, a formula in the base which remains in the new belief base retains its rank unless an inconsistency with (R1) or (R2) forces it to be changed. Following Spohn, in the case of an expansion or revision, the user must supply the rank of the added formula, as there is no way to automatically determine the degree of evidence associated with newly acquired information.

After a belief change operation, the entrenchment base must be consistent with (R1) and (R2). For example, if $A \vee B$ is a belief in the base with rank $r$ and a revision to accept $A$ with rank greater than $r$ is performed, the formula $A \vee B$ must be deleted from the base as it is now derivable from $A$ and its rank is thus automatically increased to that of $A$. Similarly, for a contraction, if $A$ is removed from a belief set, by (K-5) for every formula $B$ originally in $K$, the contracted set must contain $A \rightarrow B$. If the base originally contained $B$ but not $A \rightarrow B$, $A \rightarrow B$ must be explicitly added to the base at the appropriate rank. The details of the entrenchment modification procedure are best deferred to the discussion of our belief change algorithms, as we aim to minimize the necessary adjustments to the entrenchment as far as possible by incorporating the entrenchment revision into each of the algorithms.

## 4. A first-order AGM belief revision system

Our implementation is based on classical first-order logic, which satisfies all the requirements of the AGM approach. A belief set is represented by a finite set of first-order formulae in clausal form, and logical consequence is computed by a resolution theorem prover using OL-refutation [3]. OL-refutation is used to determine the rank of a formula by generating, in a depth-first fashion, the highest ranked OL-refutation of the negation of the formula.

Formulae in the database are represented in conjunctive normal form for use by the resolution theorem prover. For each formula in the database, the system maintains both the conjunctive normal form of the formula and that of its negation. The clausal form of the negation is needed by the refutation procedure when querying the formula, and this cannot be calculated directly from the clausal form of the formula because of information lost in Skolemization. (In particular, it is not possible in general to ascertain whether universally quantified variables appeared within the scope of existentially quantified variables.)

There is no sound and complete decision procedure for first-order derivability. OL-refutation guarantees soundness and completeness of the inference method, but does not guarantee termination. Our system provides two ways of dealing with this problem. The first option is to allow the search for a refutation to proceed indefinitely. The second allows the user to place a limit on the number of times each clause can be used as a side clause in a resolution proof. Once this limit is reached, a warning message is printed and that clause is not used again in the rest of the proof. This guarantees termination at the cost of completeness. The limit on the number of uses of each clause is a parameter which can be adjusted dynamically; used in this way, our theorem prover would then have a depth-first iterative deepening search strategy.

### 4.1 Determination of rank

A query to the system of the form ‘is $\alpha$ believed?’ is interpreted as a request to determine the rank of $\alpha$. If $\alpha$ is currently believed, it has a non-zero rank, otherwise its rank is zero. The rank of $\alpha$ is calculated by using a modified OL-resolution theorem prover, which finds proofs of $\alpha$ and associates with each proof a rank, defined as the rank of the lowest ranked formula involved in the proof. This is a lower bound for the rank of $\alpha$. Subsequent attempts to prove $\alpha$ are terminated as soon as a formula of equal or lower rank is used. When all proofs of $\alpha$ have been found, the greatest lower bound of the ranks is returned as the rank of $\alpha$. If no proof is found, the rank of $\alpha$ is zero, indicating that $\alpha$ is not a current belief.

Suppose we use **proof**(Γ, *r*, α) to denote the use of OL-refutation to find a proof of the formula α using only the formulae from the base Γ of rank at least *r*, i.e. using only the formulae in $\overline{\Gamma_r}$ as defined in section 3.1. Assume also that successive calls to this procedure return all the proofs of α in some arbitrary order. The algorithm (in pseudo-code) to find the rank of the formula α in the base Γ is:

```
rank(Γ, α)
  r := 0
  while p := proof(Γ, r+1, α) do   { more proofs exist }
    r := rank(p)
  return(r)
```

Note that our OL-refutation procedure is based on depth-first search and thus is not guaranteed to produce the highest ranked proofs first. An alternative here would be to use an OL-refutation procedure based on breadth-first search, which is guaranteed to find the highest ranked proof first, although this would be far less space efficient than our method.

## 4.2 Expansion

The expansion operator **expand**(Γ,α,*newrank*) is implemented as follows, where Γ is the base of the belief set, α is the belief being added to the base, and *newrank* is the intended rank of α. Here it is assumed that **update**(Γ, α, *newrank*) either adds the formula α to the base Γ with rank *newrank* if α is not already contained in the base, or changes the rank of α to *newrank* if α is already in the base, and that **delete**(Γ, α) deletes α from the base Γ.

```
expand(Γ,α,newrank)
  if rank(Γ, ¬α) > 0   { α is inconsistent with Γ }
    return(Γ)
  else
    oldrank := rank(Γ, α)
    if newrank ≤ oldrank
      return(Γ)
    else
      Δ := update(Γ, α, newrank)
      for each β ∈ Γ such that
          oldrank ≤ rank(Γ, β) ≤ newrank do
        if proof(Δ − {β}, rank(Γ, β), β)
          Δ := delete(Δ, β)
      return(Δ)
```

Some comments on this algorithm are in order. First, when the base is to be expanded by a formula, the formula is tested for logical consistency with the current base and the expansion rejected if it is not consistent. This disagrees with the AGM definitions, but we justify this on the basis that a user would not wish to have an inconsistent belief set. Second, if the formula is consistent with the belief set, its rank is determined for consistency with the rest of the entrenchment. The given new rank *newrank* is taken to indicate further evidence for α, so if α already has rank at least *newrank*, no change to Γ is made. If the rank of α in Γ is less than *newrank*, α is explicitly added to the base with rank *newrank* and any changes necessary to the base are made. The rank of any formula β in Γ with rank at least the old rank of α but not greater than *newrank* may be affected by the change in rank of α, in particular, if β is derivable with the same or higher rank from the other formulae in the new base after α has been added. In this case, β is redundant and may be deleted. Note that the calls to **rank**(Γ, β) in the loop of the procedure are merely lookups of the stored rank of β in the old base Γ. Note also that for each iteration of the loop, the test for whether β is derivable uses the base which may have been reduced after previous iterations of the loop have deleted some formulae. This does not affect the end result because a formula γ is deleted only if it is derivable from the formulae in the new base of at least the same rank, so β is derivable from the new base with some rank if it is also derivable from the new base with the same rank after γ has been deleted. This means that the expansion algorithm does not need a copy of the original base Γ but instead can repeatedly update the one database.

## 4.3 Contraction

We now present the algorithm **contract**(Γ, α) for contraction of the base Γ by the formula α. Assume *n* is the rank of the theorems.

```
contract(Γ, α)
  if rank(Γ, α) = n        { α is a theorem }
    return(Γ)
  else
    Δ := Γ
    oldrank := rank(Γ, α)
    for each β ∈ Γ such that
        rank(Γ, β) ≤ oldrank do
      if not(proof(Δ, oldrank+1, α ∨ β))
      { (C–) condition }
        r := rank(Δ, β)
        Δ := delete(Δ, β)
        if r < oldrank or
            not(proof(Δ, oldrank+1, α → β))
          Δ := update(Δ, α → β, r)
    return(Δ)
```

If an attempt is made to contract a theorem, the attempt is rejected in accordance with the AGM definitions. A change is made to the base if α is derivable from the base. In this case, each member β of the base is checked to see whether β is deleted from the base, and further, if α → β is to be added to the base. By the (C–) condition, β is removed from the base if α = α ∨ β. The formula α → β needs to be added to the base if it will not be derivable in the new base; its rank is that of β in the original base. The test for whether α → β is derivable in the new base can be greatly simplified. In the case where $\beta$ is ranked less than $\alpha$, given that $\alpha = \alpha \vee \beta$, $\alpha \to \beta$ must be ranked at the same level as $\beta$ by (EE3), hence $\alpha \to \beta$ cannot be derivable after $\beta$ is deleted since the original base $\Gamma$ is not redundant. The other case is where $\alpha$, $\beta$ and $\alpha \vee \beta$ are all ranked the same in $\Gamma$. In this case $\alpha \to \beta$ must be added to the base with that rank if it is not already derivable from the set of higher ranked formulae. Note that this algorithm repeatedly tests the ranks of formulae in $\Delta$ rather than in the original $\Gamma$ as the (C–) condition requires. Again, as with expansion, this does not affect the result because of the nature of the tests performed. The first test is for whether there is a proof of $\alpha \vee \beta$ using only formulae ranked higher than $\alpha$; the second is for whether there is a proof of $\alpha \to \beta$ using the same set of formulae. Since none of these formulae are affected by earlier iterations of the loop, this set of formulae is the same in each $\Delta$ as in the original $\Gamma$.

The replacement of $\beta$ by $\alpha \to \beta$ is necessary because of the AGM postulate (K-5), $K \subseteq (K^-_\alpha)^+_\alpha$, from which we can derive that if $\beta \in K$ then $K^-_\alpha \vdash \alpha \to \beta$ (if $\beta \in K$, $K^-_\alpha \cup \{\alpha\} \vdash \beta$ by (K-5), so $K^-_\alpha \vdash \alpha \to \beta$ by the deduction theorem). The replacement is redundant if $\alpha \to \beta \in K$. Note that the contraction of $\alpha$ is successful even if $\alpha$ is not contained in the base: in this case, at least one formula from the base contributing to each proof of $\alpha$ is removed. Note that the appropriate rank for the added formula $\alpha \to \beta$ is that of $\beta$ in the original base $\Gamma$ because, given that $\alpha \to \beta$ is not provable from the set of formulae ranked higher than $\beta$, its rank in $\Gamma$ is the same as that of $\beta$ in $\Gamma$.

It can be shown that the resulting set of formulae is a base for the theory obtained by applying the AGM contraction operator to the closure of the base:

$$Cn(\textbf{contract}(\Gamma, \alpha)) = Cn(\Gamma)^-_\alpha$$

Thus our implementation of contraction is correct.

### 4.4 Revision

We now present the algorithm **revise**($\Gamma$, $\alpha$, *newrank*) for revising a belief base $\Gamma$ by a formula $\alpha$ with rank *newrank*. Assume $n$ is the rank of the theorems.

```
revise(Γ, α, newrank)
  if rank(Γ, α) = n or rank(Γ, ¬α) = n
  { α either a theorem or contradiction }
    return(Γ)
  else
    Δ := Γ
    oldrank := rank(Γ, ¬α)
    for each β ∈ Γ such that rank(Δ, β) ≤ oldrank do
      if not(proof(Δ, oldrank+1, α → β))
      { (C*) condition }
        Δ := delete(Δ, β)
    Δ := expand(Δ, α, newrank)
    return(Δ)
```

If $\alpha$ is a theorem, the revision is trivial and no change is made to the base. If $\alpha$ is inconsistent, the operation is disallowed for reasons similar to those for disallowing inconsistent expansions. Otherwise we remove from the base any formula $\beta$ such that $\neg\alpha$ is of rank equal to $\alpha \to \beta$ in the original base. It suffices to consider only those $\beta$ ranked less than or equal to $\neg\alpha$ because if $\neg\alpha < \beta$ then $\neg\alpha < \alpha \to \beta$, so $\beta$ remains in the revised set. Finally, the set of formulae remaining in the base is expanded by $\alpha$. Note that, again, the repeated calls to **proof** use only the formulae ranked higher than $\neg\alpha$, and this set does not change with repetitions of the loop.

This algorithm is also correct in the sense that:

$$Cn(\textbf{revise}(\Gamma, \alpha, newrank)) = Cn(\Gamma)^*_\alpha$$

if $\alpha$ is consistent.

## 5. Examples

### 5.1 Interface to the system

Input to the program consists of commands to either expand, contract or revise the theory base, or to ask queries, conditional queries or entrenchment queries of the system. Formulae use the following symbols:

| | |
|---|---|
| ~ | negation |
| \| | disjunction |
| & | conjunction |
| -> | implication |
| <- | reverse implication |
| <-> | double implication |
| E(*X*)[*f*] | variable *X* is existentially quantified in formula *f* |
| A(*X*)[*f*] | variable *X* is universally quantified in formula *f* |

Variables begin with upper case letters; predicate and function symbols begin with lower case. Parentheses are used to disambiguate formulae: otherwise the usual precedences apply. Free variables are implicitly universally quantified in database formulae, and existentially quantified in queries, as usual with resolution theorem provers.

The following syntax is used for commands:

| | |
|---|---|
| *n*: *f* | expansion by formula *f*, setting rank(*f*) = *n* |
| !*f* | contraction by formula *f* |
| *n***f* | revision by formula *f*, setting rank(*f*) = *n* |
| ?*f* | query formula *f* |
| ?*f* => *g* | query conditional *f* => *g* |
| ?*f* ? *g* | query entrenchment *f* ? *g* |
| ?? | print current theory base |
| $*n* | set maximum number of times a clause can be used in a proof to *n* (set *n* = 0 for no limit) |

The conditional query "?*f* => *g*" is true with respect to a belief set $K$ if $g \in K^*_f$; $K$ is left unchanged.

The following section shows the operation of the expansion, contraction and revision algorithms. The system’s responses are indented with ‘>>>’.

## 5.2 Expansion, contraction and revision

```
100: A(X) [p(X) & q(X) -> r(X)]
10: r(a)
20: p(a)
30: q(a)
>>> Deleting r(a) from base : derived rank = 20
```

Here r(a) is derivable from p(a) with rank 20, q(a) with rank 30 and the rule with rank 100. Hence the rank of r(a) is now 20, and r(a) is deleted from the base as it is redundant.

```
40: p(a) | r(a)
??
>>> Complete database:
>>> 100 : All(X) [p(X) & q(X) -> r(X)]
>>>  40 : p(a) | r(a)
>>>  30 : q(a)
>>>  20 : p(a)
? r(a)
>>> yes : rank = 30
? r(a) ? r(a) | p(a)
>>> r(a) < r(a) | p(a)
```

Note that the addition of p(a) $\vee$ r(a) means that r(a) now has rank 30 because it is derivable from q(a), p(a) $\vee$ r(a) and the rule. The entrenchment relation between r(a) and r(a) $\vee$ p(a) is queried: r(a) < r(a) $\vee$ p(a).

```
! r(a)
??
>>> Complete database:
>>> 100 : All(X) [p(X) & q(X) -> r(X)]
>>>  40 : p(a) | r(a)
>>>  30 : (r(a)) -> (q(a))
>>>  20 : p(a)
```

Now the database is contracted by r(a). Because r(a) < r(a) $\vee$ p(a), p(a) remains in the contracted belief set by the (C–) condition, but the belief q(a) with rank 30 (so that r(a) has the same rank as r(a) $\vee$ q(a)) is replaced by r(a) $\rightarrow$ q(a), as required by (K-5).

```
40* ˜r(a)
>>> Deleting p(a) from base : derived rank = 40
>>> Deleting (r(a)) -> (q(a)) from base :
>>>    derived rank = 40
??
>>> Complete database:
>>> 100 : All(X) [p(X) & q(X) -> r(X)]
>>>  40 : p(a) | r(a)
>>>  40 : ˜r(a)
```

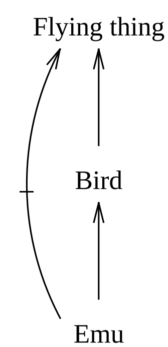
Finally the belief set is revised by $\neg$r(a) with rank 40. The formula $\neg$r(a) is added explicitly to the base with rank 40. The rank of p(a) is now implicitly increased to 40 and this formula deleted because p(a) is derivable from $\neg$r(a) and p(a) $\vee$ r(a).

## 5.3 Inheritance networks

One standard problem in nonmonotonic reasoning is that of reasoning about inheritance networks [2], which we use to illustrate conditional queries. In an inheritance network, objects are described by their place in a hierarchy of types, where more specific types inherit information from more general types. The inheritance network must be able to deal with contradictions in the inherited information, and this is generally done by allowing information from more specific types to override the information inherited from a more general type. The classic example is:

Birds fly.
Emus are birds.
Emus don’t fly.

An inheritance network is often represented by a directed acyclic graph, as shown below. Directed links connect subtypes to their supertypes, so an arrow from X to Y represents the fact that generally X’s are Y’s. The crossed arrows represent negative links, so a crossed arrow from X to Y represents the fact that X’s are not generally Y’s. So the above example would be represented as:



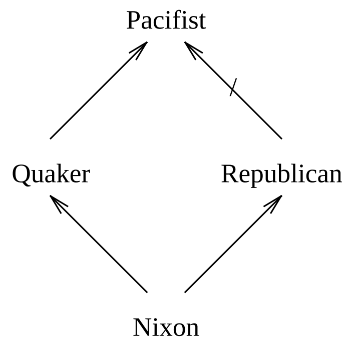

To encode this example in first-order logic, we write the inheritance rules as universal generalizations, and use the rank of a formula to encode the relative strengths of the rules. If more specific information is to override general information, the more specific rule should have a higher rank than the general rule.

```
10: All(X) [bird(X) -> flying_thing(X)]
20: All(X) [emu(X) -> bird(X)]
20: All(X) [emu(X) -> ˜flying_thing(X)]
? emu(X) => flying_thing(X)
>>> no
? emu(X) => ˜flying_thing(X)
>>> yes
```

The two requests, "Are emus flying things?" and "Are emus not flying things?" are answered correctly. The conditional query has the property that the base is not affected by the query, whereas the revision operator would remove rules which were inconsistent with current facts.

The next simple example, the Nixon diamond, provides an example where an object is an instance of two types with conflicting properties. The desired conclusion is that the system should be agnostic about both properties.

Republicans are not pacifists.
Quakers are pacifists.
Nixon is a Republican.
Nixon is a Quaker.




We then want to know whether or not Nixon is a pacifist. The example as stated would be encoded as follows:

```
10: All(X) [quaker(X) -> pacifist(X)]
10: All(X) [republican(X) -> ˜pacifist(X)]
? republican(n) & quaker(n) => pacifist(n)
>>> no
? republican(n) & quaker(n) => ˜pacifist(n)
>>> no
```

Note that in the absence of additional information to break the conflict, it is impossible to conclude that Nixon is or is not a pacifist. Now suppose we were given further information, that it is more likely for a Quaker to be a non-pacifist than a Republican to be a pacifist. Then we encode this information by giving the more reliable rule a higher rank than the other, and the example becomes:

```
10: All(X) [quaker(X) -> pacifist(X)]
20: All(X) [republican(X) -> ˜pacifist(X)]
? republican(n) & quaker(n) => ˜pacifist(n)
>>> yes
```

The system correctly derives that Nixon is not a pacifist.

## 6. Conclusion

We have developed a first-order logic belief revision system which satisfies all of the AGM postulates for any finitely representable belief state. The system uses a resolution theorem prover together with specific algorithms for the efficient computation of expansion, contraction and revision operations. These algorithms are based on the interpretation of a partially specified entrenchment as standing for a single ‘most conservative’ entrenchment, which is well motivated both on grounds of evidence and by the close relationship of belief revision to nonmonotonic reasoning. In future research we intend to use the system to implement further aspects of nonmonotonic reasoning which require the use of reasoning with equality.

## Acknowledgements

The first author is supported by an Australian Postgraduate Research Award. This research is supported by an Australian Research Council Small Grant.

## References

[1] Alchourrón, C.E., Gärdenfors, P. & Makinson, D. (1985) On the Logic of Theory Change: Partial Meet Contraction and Revision Functions, *Journal of Symbolic Logic*, **50**, pp. 510-530.
[2] Brewka, G. (1991) *Nonmonotonic Reasoning: Foundations of Common Sense*, Cambridge University Press, Cambridge.
[3] Chang, C.L. & Lee, R.C.T. (1973) *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, NY.
[4] Dixon, S.E. (1993) A Finite Base Belief Revision System, *Proceedings of the Sixteenth Australian Computer Science Conference*, pp. 445-451.
[5] Dixon, S.E. & Foo, N.Y. (1993) Connections Between the ATMS and AGM Belief Revision, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*.
[6] Gärdenfors, P. (1988) *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, Bradford Books, MIT Press, Cambridge, MA.
[7] Gärdenfors, P. & Makinson, D. (1988) Revisions of Knowledge Systems Using Epistemic Entrenchment, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pp. 83-95.
[8] Gärdenfors, P. (1990) Belief Revision and Nonmonotonic Logic: Two Sides of the Same Coin?, *Proceedings of the Ninth European Conference on Artificial Intelligence*, pp. 768-773.
[9] Gärdenfors, P. (1991) Nonmonotonic Inferences Based on Expectations: A Preliminary Report, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 585-590.
[10] Nebel, B. (1991) Belief Revision and Default Reasoning: Syntax-Based Approaches, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 417-428.
[11] Peppas, P., Foo, N.Y. & Wobcke, W.R. (1991) Events as Theory Operators, *Proceedings of the First World Conference on the Fundamentals of Artificial Intelligence*, pp. 413-426.
[12] Peppas, P. & Wobcke, W.R. (1992) On the Use of Epistemic Entrenchment in Reasoning about Action, *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp. 403-407.
[13] Poole, D. (1988) A Logical Framework for Default Reasoning, *Artificial Intelligence*, **36**, pp. 27-47.
[14] Spohn, W. (1988) Ordinal Conditional Functions: A Dynamic Theory of Epistemic States, in *Causation in Decision, Belief Change, and Statistics, II*, Harper, W.L. & Skyrms, B. (Eds), Kluwer, Dordrecht.
[15] Williams, M. (1992) Two Operators for Theory Base Change, *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence*, pp. 259-265.
[16] Wobcke, W.R. (1992) On the Use of Epistemic Entrenchment in Nonmonotonic Reasoning, *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp. 324-328.
[17] Wobcke, W.R. (1992) A Belief Revision Approach to Nonmonotonic Reasoning, *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence*, pp. 278-283.