Proc., NRAC-2011

On the Use of Epistemic Ordering Functions as Decision Criteria for Automated and Assisted Belief Revision in SNePS
(Preliminary Report)

Ari I. Fogel and Stuart C. Shapiro
University at Buffalo, The State University of New York, Buffalo, NY
{arifogel,shapiro}@buffalo.edu

Abstract

We implement belief revision in SNePS based on a user-supplied epistemic ordering of propositions. We provide a decision procedure that performs revision completely automatically when given a well preorder. We also provide a decision procedure for revision that, when given a total preorder, simulates a well preorder by making a minimal number of queries to the user when multiple propositions within a minimally-inconsistent set are minimally-epistemically-entrenched. The first procedure uses $O(|\Sigma|)$ units of space, and completes within $O(|\Sigma|^2 \cdot s_{max})$ units of time, where Σ is the set of distinct minimally-inconsistent sets, and s_{max} is the number of propositions in the largest minimally-inconsistent set. The second procedure uses $O(|\Sigma|^2 \cdot s_{max}^2)$ space and $O(|\Sigma|^2 \cdot s_{max}^2)$ time. We demonstrate how our changes generalize previous techniques employed in SNePS.

1 Introduction

1.1 Belief Revision

Several varieties of belief revision have appeared in the literature over the years. AGM revision typically refers to the addition of a belief to a belief set, at the expense of its negation and any other beliefs supporting its negation [Alchourron *et al.*, 1985]. Removal of a belief and beliefs that support it is called *contraction*. Alternatively, revision can refer to the process of resolving inconsistencies in a contradictory knowledge base, or one *known to be inconsistent* [Martins and Shapiro, 1988]. This is accomplished by removing one or more beliefs responsible for the inconsistency, or *culprits*. This is the task with which we are concerned. In particular, we have devised a means of automatically resolving inconsistencies by discarding the least-preferred beliefs in a belief base, according to some *epistemic ordering* [Gärdenfors, 1988; Williams, 1994; Gärdenfors and Rott, 1995].

> The problem of belief revision is that logical considerations alone do not tell you which beliefs to give up, but this has to be decided by some other means. What makes things more complicated is that beliefs in a database have logical consequences. So when giving up a belief you have to decide as well which of its *consequences* to retain and which to retract... [Gärdenfors and Rott, 1995]

In later sections we will discuss in detail how to make a choice of belief(s) to retract when presented with an inconsistent belief set.

AGM Paradigm

In [Gärdenfors, 1982; Alchourron *et al.*, 1985], operators and rationality postulates for *theory change* are discussed. In general, any operator that satisfies those postulates may be thought of as an AGM operation.

Let $Cn(A)$ refer to the closure under logical consequence of a set of propositions A. A *theory* is defined to be a set of propositions closed under logical consequence. Thus for any set of propositions A, $Cn(A)$ is a theory. It is worth noting that theories are infinite sets. [Alchourron *et al.*, 1985] discusses operations that may be performed on theories. *Partial meet contraction and revision*. defined in [Alchourron *et al.*, 1985], satisfy all of the postulates for a rational contraction and revision operator, respectively.

Theory Change on Finite Bases

> It is widely accepted that agents, because of their limited resources, believe some but by no means all of the logical consequences of their beliefs. [Lakemeyer, 1991]

A major issue with the AGM paradigm it *tends to* operate on and produce infinite sets (theories). A more practical model would include operations to be performed on finite belief sets, or *belief bases*. Such operators would be useful in supporting computer-based implementations of revision systems [Williams, 1994].

It has been argued that The AGM paradigm uses a *coherentist* approach[1] [Gärdenfors, 1989], in that all beliefs require some sort of external justification. On the other hand, finite-base systems are said to use a foundationalist approach, wherein some beliefs indeed have their own epistemic standing, and others can be derived from them. SNePS, as we shall see, uses the finite-base foundationalist approach.

[1]It has also been argued otherwise [Hansson and Olsson, 1999]

Epistemic Entrenchment

Let us assume that the decision on which beliefs to retract from a belief base is made is based on the relative importance of each belief, which is called its degree of *epistemic entrenchment* [Gärdenfors, 1988]. Then we need an ordering $\leqslant$ with which to compare the entrenchment of individual beliefs. Beliefs that are less entrenched are preferentially discarded during revision over beliefs that are more entrenched. An epistemic entrenchment ordering is used to uniquely determine the result of AGM contraction. Such an ordering is a noncircular total preorder (that satisfies certain other postulates) on *all propositions*.

Ensconcements

Ensconcements, introduced in [Williams, 1994], consist of a set of forumlae together with a total preorder on that set. They can be used to construct epistemic entrenchment orderings, and determine theory base change operators.

Safe Contraction

In [Alchourron and Makinson, 1985], the operation *safe contraction* is introduced. Let $<$ be a non-circular relation over a belief set A. An element a of A is *safe* with respect to x iff a is not a minimal element of any minimal subset B of A such that $x \in Cn(B)$. Let A/x be the set of all elements of A that are safe with respect to x. Then the safe contraction of A by x, denoted $A \dot{-}_S x$, is defined to be $A \cap Cn(A/x)$.

Assumption-based Truth Maintenance Systems

In an assumption-based truth maintenance system (ATMS), the system keeps track of the assumptions (base beliefs) underlying each belief [de Kleer, 1986]. One of the roles of an conventional TMS is to keep the database contradiction-free. In an assumption-based ATMS, contradictions are removed as they are discovered. When a contradiction is detected in an ATMS, then there will be one or more minimally-inconsistent sets of assumptions underlying the contradiction. Such sets are called *no-goods*. [Martins and Shapiro, 1988] presented SNeBR, an early implementation of an ATMS that uses the logic of SNePS. In that paper, sets of assumptions supporting a belief are called *origin sets*. They correspond to *antecedents* of a *justification* from [de Kleer, 1986]. The focus of this paper is modifications to the modern version of SNeBR.

Kernel Contraction

In [Hansson, 1994], the operation *kernel contraction* is introduced. A *kernel set* $A \perp\!\!\!\perp \alpha$ is defined to be the set of all minimal subsets of A that imply α. A kernel set is like a set of *origin sets* from [Martins and Shapiro, 1988]. Let σ be an *incision function* for A. Then for all α, $\sigma(A \perp\!\!\!\perp \alpha) \subseteq \cup(A \perp\!\!\!\perp \alpha)$, and if $\emptyset \neq X \in A \perp\!\!\!\perp \alpha$, then $X \cap \sigma(A \perp\!\!\!\perp \alpha) \neq \emptyset$. The *kernel contraction* of A by α based on σ, denoted $A \sim_\sigma \alpha$, is equal to $A \backslash \sigma(A \perp\!\!\!\perp \alpha)$.

Prioritized Versus Non-Prioritized Belief Revision

> In the AGM model of belief revision [Alchourron *et al.*, 1985] ...the input sentence is always accepted. This is clearly an unrealistic feature, and ...several models of belief change have been proposed in which no absolute priority is assigned to the new information due to its novelty. ...One way to construct non-prioritized belief revision is to base it on the following two-step process: First we decide whether to accept or reject the input. After that, if the input was accepted, it is incorporated into the belief state [Hansson, 1999].

Hansson goes on to describe several other models of nonprioritized belief revision, but they all have one unifiying feature distinguishing them from prioritized belief revision: the input, i.e. the RHS argument to the revision operator, is not always accepted. To reiterate: *Prioritized belief revision* is revision in which the proposition by which the set is revised is always present in the result (as long as it is not a contradiction). *Non-prioritized belief revision* is revision in which the RHS argument to the revision operator is not always present in the result (even if it is not a contradiction).

The closest approximation from Hansson's work to our work is the operation of *semi-revision* [Hansson, 1997]. Semi-revision is a type of non-prioritized belief revision that may be applied to belief bases.

1.2 SNePS

Description of the System

"SNePS is a logic-, frame-, and network- based knowledge representation, reasoning, and acting system. Its logic is based on Relevance Logic [Shapiro, 1992], a paraconsistent logic (in which a contradiction does not imply anything whatsoever) [Shapiro and Johnson, 2000]."

SNeRE, the SNePS Rational Engine, provides an acting system for SNePS-based agents, whose beliefs must change to keep up with a changing world. Of particular interest is the *believe* action, which is used to introduce beliefs that take priority over all other beliefs at the time of their introduction.

Belief Change in SNePS

Every belief in a SNePS knowledge base (which consists of a belief base and all currently-known derived propostions therefrom) has one or more *support sets*, each of which consists of an *origin tag* and an *origin set*. The origin tag will identify a belief as either being introduced as a hypothesis, or derived (note that it is possible for a belief to be both introduced as a hypothesis and derived from other beliefs). The origin set contains those *hypotheses* that were used to derive the belief. In the case of the origin tag denoting a hypothesis, the corresponding origin set would be a singleton set containing only the belief itself. The contents of the origin set of a derived belief are computed by the implemented rules of inference at the time the inference is drawn [Martins and Shapiro, 1988; Shapiro, 1992].

The representation of beliefs in SNePS lends itself well to the creation of processes for contraction and revision. Specifically, in order to contract a belief, one must merely remove at least one hypothesis from each of its origin sets. Similarly, prioritized revision by a belief b (where $\neg b$ is already believed) is accomplished by removing at least one belief from each origin set of $\neg b$. Non-prioritized belief revision under this paradigm is a bit more complicated. We discuss both types of revision in more detail in §2.

SNeBR

SNeBR, The SNePS Belief Revision subsystem, is responsible for resolving inconsistencies in the knowledge base as they are discovered. In the current release of SNePS (version 2.7.1), SNeBR is able to *automatically* resolve contradictions under a limited variety of circumstances [Shapiro and The SNePS Implementation Group, 2010, 76]. Otherwise "assisted culprit choosing" is performed, where the user must manually select culprits for removal. After belief revision is performed, the knowledge base might still be inconsistent, but every *known* derivation of an inconsistency has been eliminated.

2 New Belief Revision Algorithms

2.1 Problem Statement

Nonprioritized Belief Revision

Suppose we have a knowledge base that is not known to be inconsistent, and suppose that at some point we add a contradictory belief to that knowledge base. Either that new belief directly contradicts an existing belief, or we derive a belief that directly contradicts an existing one as a result of performing forward and/or backward inference on the new belief. Now the knowledge base is known to be inconsistent. We will refer to the contradictory beliefs as p and $\neg p$

Since SNePS tags each belief with one or more origin sets, or sets of supporting hypotheses, we can identify the underlying beliefs that support each of the two contradictory beliefs. In the case where p and $\neg p$ each have one origin set, OS_p and $OS_{\neg p}$ respectively, we may resolve the contradiction by removing at least one hypothesis from $OS_p \cup OS_{\neg p}$. We shall refer to such a union as a *no-good*. If there are m origin sets for p, and n origin sets for $\neg p$, then there will be at most $m \times n$ distinct no-goods (some unions may be duplicates of others). To resolve a contradiction in this case, we must retract at least one hypothesis from each no-good (Sufficiency).

We wish to devise an algorithm that will select the hypotheses for removal from the set of no-goods. The first priority will be that the hypotheses selected should be minimally-epistemically-entrenched (Minimal Entrenchment) according to some total preorder $\leqslant$. Note that we are not referring strictly to an AGM entrenchment order, but to a total preorder on the set of hypotheses, without regard to the AGM postulates. The second priority will be not to remove any more hypotheses than are necessary in order to resolve the contradiction (Information Preservation), while still satisfying priority one.

Prioritized Belief Revision

The process of Prioritized Belief Revision in SNePS occurs when a contradiction is discovered after a belief is asserted explicitly using the *believe* act of SNeRE. The major difference here is that a subtle change is made to the entrenchment ordering $\leqslant$. If $\leqslant_{nonpri}$ is the ordering used for nonprioritized belief revision, then for prioritized belief revision we use an ordering $\leqslant_{pri}$ as follows:

Let P be the set of beliefs asserted by a *believe* action. Then

$\forall e_1, e_2[e_1 \in P \land e_2 \notin P \rightarrow \neg(e_1 \leqslant_{pri} e_2) \land e_2 \leqslant_{pri} e_1]$

$\forall e_1, e_2[e_1 \notin P \land e_2 \notin P \rightarrow (e_1 \leqslant_{pri} e_2 \leftrightarrow e_1 \leqslant_{nonpri} e_2)]$

$\forall e_1, e_2[e_1 \in P \land e_2 \in P \rightarrow (e_1 \leqslant_{pri} e_2 \leftrightarrow e_1 \leqslant_{nonpri} e_2)]$

That is, a proposition asserted by a *believe* action takes priority over any other proposition. When either both or neither propositions being compared have been assserted by the *believe* action, then we use the same ordering as we would for nonprioritized revision.

2.2 Common Requirements for a Rational Belief Revision Algorithm

Primary Requirements

The inputs to the algorithm are:

- A set of formulae Φ: the current belief base, which is known to be inconsistent
- A total preorder $\leqslant$ on Φ: an epistemic entrenchment ordering that can be used to compare the relative desirability of each belief in the current belief base
- Minimally-inconsistent sets of formulae $\sigma_1, \dots, \sigma_n$, each of which is a subset of Φ: the no-goods
- A set $\Sigma = \{\sigma_1, \dots, \sigma_n\}$: the set of all the no-goods

The algorithm should produce a set T that satisfies the following conditions:

$(EE_{SNePS}1)$ $\forall\sigma[\sigma \in \Sigma \rightarrow \exists\tau[\tau \in (T \cap \sigma)]$ (Sufficiency)

$(EE_{SNePS}2)$ $\forall\tau[\tau \in T \rightarrow \exists\sigma[\sigma \in \Sigma \land \tau \in \sigma \land \forall w[w \in \sigma \rightarrow \tau \leqslant w]]]$ (Minimal Entrenchment)

$(EE_{SNePS}3)$ $\forall T'[T' \subset T \rightarrow \neg\forall\sigma[\sigma \in \Sigma \rightarrow \exists\tau[\tau \in (T' \cap \sigma)]]]$ (Information Preservation)

Condition $(EE_{SNePS}1)$ states that T contains at least one formula from each set in Σ. Condition $(EE_{SNePS}2)$ states that every formula in T is a minimally-entrenched formula of some set in Σ. Condition $(EE_{SNePS}3)$ states that if any formula is removed from T, then Condition $(EE_{SNePS}1)$ will no longer hold. In addition to the above conditions, our algorithm must terminate on all possible inputs, i.e. it must be a decision procedure.

Supplementary Requirement

In any case where queries must be made of the user in order to determine the relative epistemic ordering of propositions, the number of such queries must be kept to a minimum.

2.3 Implementation

We present algorithms to solve the problem as stated:

Where we refer to $\leqslant$ below, we are using the *prioritized* entrenchment ordering from §2.1. In the case of nonprioritized revision we may assume that $P = \varnothing$

Using a well preorder

Let $\leq_{\leqslant}$ be the output of a function f whose input is a total preorder $\leqslant$, such that $\leq_{\leqslant} \subseteq \leqslant$ The idea is that f creates the well preorder $\leq_{\leqslant}$ from $\leqslant$ by removing some pairs from the total preorder $\leqslant$. Note that in the case where $\leqslant$ is already a well preorder, $\leq_{\leqslant} = \leqslant$. Then we may use Algorithm 1 to solve the problem.

Algorithm 1 Algorithm to compute T given a well preorder

```
Input: $\Sigma, \leq_{\leqslant}$
Output: $T$
1: $T \Leftarrow \varnothing$
2: for all $(\sigma \in \Sigma)$ do
3:     Move minimally entrenched belief in $\sigma$ to first position
       in $\sigma$, using $\leq_{\leqslant}$ as a comparator
4: end for
5: Sort elements of $\Sigma$ into descending order of the values of
   the first element in each $\sigma$ using $\leq_{\leqslant}$ as a comparator
6: $AddLoop$ :
7: while $(\Sigma \neq \varnothing)$ do
8:     $currentCulprit \Leftarrow \sigma_{1_1}$
9:     $T \Leftarrow T \cup \{currentCulprit\}$
10:    $DeleteLoop$ :
11:    for all $(\sigma_{current} \in \Sigma)$ do
12:        if $(currentCulprit \in \sigma_{current})$ then
13:            $\Sigma \Leftarrow \Sigma \backslash \sigma_{current}$
14:        end if
15:    end for
16: end while
17: return $T$
```

Using a total preorder

Unfortunately it is easy to conceive of a situation in which the supplied entrenchment ordering is a total preorder, but not a well preorder. For instance, let us say that, when reasoning about a changing world, propositional fluents (propositions that are only true of a specific time or situation) are abandoned over non-fluent propositions. It is not clear then how we should rank two distinct propositional fluents, nor how to rank two distinct non-fluent propositions. If we can arbitrarily specify a well preorder that is a subset of the total preorder we are given, then algorithm 1 will be suitable. Otherwise, we can simulate a well order $\leq$ through an iterative construction by querying the user for the unique minimally-entrenched proposition of a particular set of propositions at appropriate times in the belief-revision process. Algorithm 2 accomplishes just this.

Algorithm 2 Algorithm to compute T given a total preorder

```
Input: $\Sigma, \leqslant$
Output: $T$
1: $T \Leftarrow \varnothing$
2: $MainLoop$:
3: loop
4:     $ListLoop$:
5:     for all $(\sigma_i \in \Sigma, 1 \leqslant i \leqslant |\Sigma|)$ do
6:         Make a list $l_{\sigma_i}$ of all minimally-entrenched propo-
           sitions, i.e. propositions that are not strictly more
           entrenched than any other, among those in $\sigma_i$, using
           $\leqslant$ as a comparator.
7:     end for
8:     $RemoveLoop$:
9:     for all $(\sigma_i \in \Sigma, 1 \leqslant i \leqslant |\Sigma|)$ do
10:        if (According to $l_{\sigma_i}$, $\sigma$ has exactly one minimally-
           entrenched proposition $p$ AND the other proposi-
           tions in $\sigma_i$ are not minimally-entrenched in any
           other no-good via an $l_{\sigma_j}$,$(1 \leqslant j \leqslant |\Sigma|, i \neq j)$) then
11:            $T \Leftarrow T \cup \{p\}$
12:            for all $(\sigma_{current} \in \Sigma)$ do
13:                if $(p \in \sigma_{current})$ then
14:                    $\Sigma \Leftarrow \Sigma \backslash \sigma_{current}$
15:                end if
16:            end for
17:            if $(\Sigma = \varnothing)$ then
18:                return $T$
19:            end if
20:        end if
21:    end for
22:    $ModifyLoop$:
23:    for all $(\sigma \in \Sigma)$ do
24:        if ($\sigma$ has multiple minimally-entrenched proposi-
           tions) then
25:            query which proposition $l$ of the minimally-
               entrenched propostions is least desired.
26:            Modify $\leqslant$ so that $l$ is strictly less entrenched than
               those other propositions.
27:            break out of $ModifyLoop$
28:        end if
29:    end for
30: end loop
```

Characterization

These algorithms perform an operation similar to *incision functions* [Hansson, 1994], since they select one or more propositions to be removed from each minimally-inconsistent set. Their output seems analogous to $\sigma(\Phi \perp\!\!\!\perp (p \wedge \neg p))$, where σ is the incision function, $\perp\!\!\!\perp$ is the kernel-set operator from [Hansson, 1994], and p is a proposition. But we are actually incising Σ, the set of *known* no-goods. The known no-goods are of course a subset of all no-goods, i.e. $\Sigma \subseteq \Phi \perp\!\!\!\perp (p \wedge \neg p)$. This happens because SNeBR resolves contradictions as soon as they are discovered, rather than performing inference first to discover all possible sources of contradictions.

The type of contraction eventually performed is similar to safe contraction [Alchourron and Makinson, 1985], except that there are fewer restrictions on our epistemic ordering.

3 Analysis of Algorithm 1

3.1 Proofs of Satisfaction of Requirements by Algorithm 1

We show that Algorithm 1 satisfies the requirements established in section 2:

$(EE_{SNePS}1)$ **(Sufficiency)**
During each iteration of *AddLoop* an element τ is added to T from some $\sigma \in \Sigma$. Then each set $\sigma \in \Sigma$ containing τ is removed from Σ. The process is repeated until Σ is empty. Therefore each removed set σ in Σ contains some τ in T (Note that each σ will be removed from Σ by the end of the process). So $\forall \sigma[\sigma \in \Sigma \rightarrow \exists \tau[\tau \in (T \cap \sigma)]$. Q.E.D.

$(EE_{SNePS}2)$ **(Minimal Entrenchment)**
From lines 8-9, we see that T is comprised solely of first elements of sets in Σ. And from lines 2-4, we see that those first elements are all minimal under $\leq_{\leqslant}$ relative to the other

elements in each set. Since $\forall e_1, e_2, \leqslant [e_1 \leq_{\leqslant} e_2 \to e_1 \leqslant e_2]$, those first elements are minimal under $\leqslant$ as well. That is, $\forall \tau[\tau \in T \to \exists \sigma[\sigma \in \Sigma \wedge \tau \in \sigma \wedge \forall w[w \in \sigma \to \tau \leqslant w]]]$. Q.E.D.

$(EE_{SNePS}3)$ **(Information Preservation)**
From the previous proof we see that during each iteration of *AddLoop*, we are guaranteed that at least one set σ containing the current culprit is removed from Σ. And we know that the current culprit for that iteration is minimally-entrenched in σ. We also know from $(EE_{SNePS}2)$ that each subsequently chosen culprit will be minimally entrenched in some set. From lines 2-5 and *AddLoop*, we know that subsequently chosen culprits will be less entrenched than the current culprit. From lines 2-5, we also see that all the other elements in σ have higher entrenchment than the current culprit. Therefore subsequent culprits cannot be elements in σ. So, they cannot be used to eliminate σ. Obviously, previous culprits were also not members of σ. Therefore, if we exclude the current culprit from T, then there will be a set in Σ that does not contain any element of T. That is,
$\forall T'[T' \subset T \to \exists \sigma[\sigma \in \Sigma \wedge \neg \exists \tau[\tau \in (T' \cap \sigma)]]]$
$\therefore \forall T'[T' \subset T \to \exists \sigma[\neg\neg(\sigma \in \Sigma \wedge \neg \exists \tau[\tau \in (T' \cap \sigma)])]]$
$\therefore \forall T'[T' \subset T \to \exists \sigma[\neg(\neg(\sigma \in \Sigma) \vee \exists \tau[\tau \in (T' \cap \sigma)])]]$
$\therefore \forall T'[T' \subset T \to \exists \sigma[\neg(\sigma \in \Sigma \to \exists \tau[\tau \in (T' \cap \sigma)])]]$
$\therefore \forall T'[T' \subset T \to \neg \forall \sigma[\sigma \in \Sigma \to \exists \tau[\tau \in (T' \cap \sigma)]]]$ Q.E.D.

Decidability
We see that *DeleteLoop* is executed once for each element in Σ, which is a finite set. So it always terminates. We see that *AddLoop* terminates when Σ is empty. And from lines 8 and 13 we see that at least one set is removed from Σ during each iteration of *AddLoop*. So *AddLoop* always terminates. Lines 2-4 involve finding a minimum element, which is a decision procedure. Line 5 performs sorting, which is also a decision procedure. Since every portion of Algorithm 1 always terminates, it is a decision procedure. Q.E.D.

Supplementary Requirement
Algorithm 1 is a fully-automated procedure that makes no queries of the user. Q.E.D.

3.2 Complexity of Algorithm 1

Space Complexity
Algorithm 1 can be run completely in-place, i.e. it can use only the memory allocated to the input, with the exception of the production of the set of culprits T. Let us assume that the space needed to store a single proposition is $O(1)$ memory units. Since we only need to remove one proposition from each no-good to restore consistency, algorithm 1 uses $O(|\Sigma|)$ memory units.

Time Complexity
The analysis for time complexity is based on a sequential-procesing system. Let us assume that we implement lists as array structures. Let us assume that we may determine the size of an array in $O(1)$ time. Let us also assume that performing a comparison using $\leq_{\leqslant}$ takes $O(1)$ time. Then in lines 2-4, for each array $\sigma \in \Sigma$ we find the minimum element σ and perform a swap on two elements at most once for each element in σ. If we let s_{max} be the cardinality of the largest σ in Σ, then lines 2-4 will take $O(|\Sigma| \cdot s_{max})$ time. In line 5, we sort the no-goods' positions in Σ using their first elements as keys. This takes $O(|\Sigma| \cdot log(|\Sigma|))$ time. Lines 7-16 iterate through the elements of Σ at most once for each element in Σ. During each such iteration, a search is performed for an element within a no-good. Also, during each iteration through all the no-goods, at least one σ is removed, though this does not help asymptotically. Since the no-goods are not sorted, the search takes linear time in s_{max}. So lines 7-16 take $O(|\Sigma|^2 \cdot s_{max})$ time. Therefore, the running time is $O(|\Sigma|^2 \cdot s_{max})$ time.

Note that the situation changes slightly if we sort the no-goods instead of just placing the minimally-entrenched proposition at the front, as in lines 2-4. In this case, each search through a no-good will take $O(log(s_{max}))$ time, yielding a new total time of $O(|\Sigma| \cdot s_{max} \cdot log(s_{max}) + |\Sigma|^2 \cdot log(s_{max}))$.

4 Analysis of Algorithm 2

4.1 Proofs of Satisfaction of Requirements by Algorithm 2

We show that Algorithm 2 satisfies the requirements established in section 2:

$(EE_{SNePS}1)$ **(Sufficiency)**
Since every set of propositions must contain at least one proposition that is minimally entrenched, at least one proposition is added to the list in each iteration of *ListLoop*. In the worst case, assume that for each iteration of *MainLoop*, only either *RemoveLoop* or *ModifyLoop* do any work. We know that at least this much work is done for the following reasons: if *ModifyLoop* cannot operate on any no-good during an iteration of *MainLoop*, then all no-goods have only one minimally-entrenched proposition. So either *RemoveLoop*'s condition at line 10 would hold, or:
1. A no-good has multiple minimally-entrenched propositions, causing *ModifyLoop* to do work. This contradicts our assumption that *ModifyLoop* could not do any work during this iteration of *MainLoop*, so we set this possibility aside.
2. Some proposition p_1 is a non-minimally-entrenched proposition in some no-good σ_n, and a minimally-entrenched one in another no-good σ_m. In this case, either p_1 is removed during the iteration of *RemoveLoop* where σ_m is considered, or there is another proposition p_2 in σ_m that is not minimally-entrenched in σ_m, but is in $\sigma_{m'}$. This chaining must eventually terminate at a no-good $\sigma_{m_{final}}$ since $\leqslant$ is transitive. And the final proposition in the chain p_{final} must be the sole minimally-entrenched proposition in σ_{final}, since otherwise *ModifyLoop* would have been able to do work for this iteration of *MainLoop*, which is a contradiction. *ModifyLoop* can only do work once for each no-good, so eventually its work is finished. If *ModifyLoop* has no more work left to do, then *RemoveLoop* must do work at least once for each iteration of *MainLoop*. And in doing so, it will create a list of culprits of which each no-good contains at least one. Q.E.D.

$(EE_{SNePS}2)$ **(Minimal Entrenchment)**
Since propositions are only added to T when the condition in line 10 is satisfied, it is guaranteed that every proposition in

T is a minimally-entrenched proposition in some no-good σ.

$(EE_{SNePS}3)$ **(Information Preservation)**
From line 10, we see that when a proposition p is removed, none of the other propositions in its no-good are minimally-entrenched in any other no-good. That means none of the other propositions could be a candidate for removal. So, the only way to remove the no-good in which p appears is by removing p. So if p were not removed, then $(EE_{SNePS}1)$ would not be satisfied. Q.E.D.

Decidability
ListLoop creates lists of minimal elements of lists. This is a decision procedure since the comparator is a total preorder. From the proof of $(EE_{SNePS}1)$ above, we see that either *RemoveLoop* or $ModifyLoop$ must do work for each iteration of *MainLoop*. $ModifyLoop$ cannot operate more than once on the same no-good, because there are no longer multiple minimally-entrenched propositions in the no-good after it does its work. Nor can *RemoveLoop* operate twice on the same no-good, since the no-good is removed when $ModifyLoop$ does work. So, eventually $ModifyLoop$ has no more work to do, and at that point *RemoveLoop* will remove at least one no-good for each iteration of *MainLoop*. By lines 17-18, when the last no-good is removed, the procedure terminates. So it always terminates. Q.E.D.

Supplementary Requirement
RemoveLoop attempts to compute T each time it is run from *MainLoop*. If the procedure does not terminate within *RemoveLoop*, then we run $ModifyLoop$ on *at most one* no-good. Afterwards, we run RemoveLoop again. Since the user is only queried when the procedure cannot automatically determine any propositions to remove, we argue that this means minimal queries are made of the user. Q.E.D.

4.2 Complexity of Algorithm 2

Space Complexity
As before, let s_{max} be the cardinality of the largest no-good in Σ. In the worst case all propositions are minimally entrenched, so *ListLoop* will recreate Σ. So *ListLoop* will use $O(|\Sigma| \cdot s_{max})$ space. RemoveLoop creates a culprit list, which we stated before takes $O(|\Sigma|)$ space. ModifyLoop may be implemented in a variety of ways. We will assume that it creates a list of pairs, of which the first and second elements range over propositions in the no-goods. In this case $ModifyLoop$ uses $O(|\Sigma|^2 \cdot s_{max}^2)$ space. So the total space requirement is $O(|\Sigma|^2 \cdot s_{max}^2)$ memory units.

Time Complexity
The analysis for time complexity is based on a sequential-procesing system. For each no-good σ, in the worst case, *ListLoop* will have to compare each proposition in σ agains every other. So, for each iteration of *MainLoop*, *ListLoop* takes $O(|\Sigma| \cdot s_{max}^2)$ time. There are at most $O(s_{max})$ elements in each list created by *ListLoop*. So, checking the condition in line 10 takes $O(|\Sigma| \cdot s_{max}^2)$ time. Lines 12-16 can be executed in $O(|\Sigma| \cdot s_{max})$ time. Therefore, *RemoveLoop* takes $O(|\Sigma| \cdot s_{max}^2)$ time. We assume that all the work in lines 24-27 can be done in constant time. So, $ModifyLoop$ takes $O(|\Sigma|)$ time. We noted earlier that during each iteration of *MainLoop*, *RemoveLoop* or $ModifyLoop$ will do work. In the worst case, only one will do work each time. And they each may do work at most $|\Sigma|$ times. So the total running time for the procedure is $O(|\Sigma|^2 \cdot s_{max}^2)$.

5 Annotated Demonstrations

A significant feature of our work is that it generalizes previous published work on belief revision in SNePS [Johnson and Shapiro, 1999; Shapiro and Johnson, 2000; Shapiro and Kandefer, 2005]. The following demonstrations showcase the new features we have introduced to SNeBR, and capture the essence of belief revision as seen in the papers mentioned above by using well-specified epistemic ordering functions. The demos have been edited for formatting and clarity.

The commands *br-tie-mode auto* and *br-tie-mode manual* indicate that Algorithm 1 and Algorithm 2 should be used respectively. A *wff* is a well-formed formula. A wff followed by a period (.) indicates that the wff should be asserted, i.e. added to the knowledge base. A wff followed by an exclamation point (!) indicates that the wff should be asserted, and that forward inference should be performed on it.

Says Who?
We present a demonstration on how the source-credibility-based revision behavior from [Shapiro and Johnson, 2000] is generalized by our changes to SNeBR. The knowledge base in the demo is taken from [Johnson and Shapiro, 1999]. In the following example, the command *set-order source* sets the epistemic ordering used by SNeBR to be a lisp function that compares two propositions based on the relative credibility of their sources. Unsourced propositions are assumed to have maximal credibility. The sources, as well as their relative credibility are represented as meta-knowledge in the SNePS knowledge base. This was also done in [Johnson and Shapiro, 1999] and [Shapiro and Johnson, 2000]. The *source* function makes SNePSLOG queries to determine sources of propositions and credibility of sources, using the *askwh* and *ask* commands [Shapiro and The SNePS Implementation Group, 2010]. This allows it to perform inference in making determinations about sources.

Here we see that the nerd and the sexist make the generalizations that all jocks are not smart and all females are not smart respectively, while the holy book and the professor state that all old people are smart, and all grad students are smart respectively. Since Fran is an old female jock graduate student, there are two sources that would claim she is smart, and two that would claim she is not, which is a contradiction.

```
;;; Show origin sets
: expert
: br-mode auto
Automatic belief revision will now be automatically selected.
: br-tie-mode manual
The user will be consulted when an entrenchment tie occurs
;;; Use source credibilities as epistemic ordering criteria.
set-order source
;;; The holy book is a better source than the professor.
IsBetterSource(holybook, prof).
;;; The professor is a better source than the nerd.
IsBetterSource(prof, nerd).
;;; The nerd is a better source than the sexist.
IsBetterSource(nerd, sexist).
```

```
;;; Fran is a better source than the nerd.
IsBetterSource(fran, nerd).
;;; Better—Source is a transitive relation
all(x,y,z)({IsBetterSource(x,y), IsBetterSource(y,z)} &=>
      IsBetterSource(x,z))!
;;; All jocks are not smart.
all(x)(jock(x)=>~smart(x)). ;wff10
;;; The source of the statement 'All jocks are not smart' is the nerd
HasSource(wff10, nerd).
;;; All females are not smart.
all(x)(female(x)=>~smart(x)). ;wff12
;;; The source of the statement 'All females are not smart' is the
      sexist.
HasSource(wff12, sexist).
;;; All graduate students are smart.
all(x)(grad(x)=>smart(x)). ;wff14
;;; The source of the statement 'All graduate students are smart' is
      the professor.
HasSource(wff14, prof).
;;; All old people are smart.
all(x)(old(x)=>smart(x)). ;wff16
;;; The source of the statement 'All old people are smart' is the
      holy book.
HasSource(wff16, holybook).
;;; The source of the statement 'Fran is an old female jock who is a
      graduate student' is fran.
HasSource(and{jock(fran),grad(fran),female(fran),old(fran)},fran).
;;; The KB thus far list—asserted—wffs
wff23!: HasSource(old(fran) and female(fran) and grad(fran) and
            jock(fran),fran)  {<hyp,{wff23}>}
wff17!: HasSource(all(x)(old(x) => smart(x)),holybook){<hyp,{wff17}>}
wff16!: all(x)(old(x) => smart(x))  {<hyp,{wff16}>}
wff15!: HasSource(all(x)(grad(x) => smart(x)),prof)  {<hyp,{wff15}>}
wff14!: all(x)(grad(x) => smart(x))  {<hyp,{wff14}>}
wff13!: HasSource(all(x)(female(x) => (~smart(x))),sexist)
            {<hyp,{wff13}>}
wff12!: all(x)(female(x) => (~smart(x)))  {<hyp,{wff12}>}
wff11!: HasSource(all(x)(jock(x) => (~smart(x))),nerd) <hyp,{wff11}>}
wff10!: all(x)(jock(x) => (~smart(x)))  {<hyp,{wff10}>}
wff9!:  IsBetterSource(fran,sexist)  {<der,{wff3,wff4,wff5}>}
wff8!:  IsBetterSource(prof,sexist)  {<der,{wff2,wff3,wff5}>}
wff7!:  IsBetterSource(holybook,sexist) {<der,{wff1,wff2,wff3,wff5}>}
wff6!:  IsBetterSource(holybook,nerd)  {<der,{wff1,wff2,wff5}>}
wff5!:  all(z,y,x)({IsBetterSource(y,z),IsBetterSource(x,y)} &=>
            {IsBetterSource(x,z)})  {<hyp,{wff5}>}
wff4!:  IsBetterSource(fran,nerd)  {<hyp,{wff4}>}
wff3!:  IsBetterSource(nerd,sexist)  {<hyp,{wff3}>}
wff2!:  IsBetterSource(prof,nerd)  {<hyp,{wff2}>}
wff1!:  IsBetterSource(holybook,prof)  {<hyp,{wff1}>}
;;; Fran is an old female jock who is a graduate student (asserted
      with forward inference).
and{jock(fran),grad(fran),female(fran),old(fran)}!
wff50!: ~(all(x)(jock(x) => (~smart(x))))
            {<ext,{wff16,wff22}>,<ext,{wff14,wff22}>}
wff24!: smart(fran)  {<der,{wff16,wff22}>,<der,{wff14,wff22}>}
;;; The resulting knowledge base (HasSource and IsBetterSource omited
       for clarity)
list—asserted—wffs
wff50!: ~(all(x)(jock(x) => (~smart(x))))
            {<ext,{wff16,wff22}>, <ext,{wff14,wff22}>}
wff37!: ~(all(x)(female(x) => (~smart(x))))  {<ext,{wff16,wff22}>}
wff24!: smart(fran)  {<der,{wff16,wff22}>,<der,{wff14,wff22}>}
wff22!: old(fran) and female(fran) and grad(fran) and jock(fran)
            {<hyp,{wff22}>}
wff21!: old(fran)  {<der,{wff22}>}
wff20!: female(fran)  {<der,{wff22}>}
wff19!: grad(fran)  {<der,{wff22}>}
wff18!: jock(fran)  {<der,{wff22}>}
wff16!: all(x)(old(x) => smart(x))  {<hyp,{wff16}>}
wff14!: all(x)(grad(x) => smart(x))  {<hyp,{wff14}>}
```

We see that the statements that all jocks are not smart and that all females are not smart are no longer asserted at the end. These statements supported the statement that Fran *is not* smart. The statements that all old people are smart and that all grad students are smart supported the statement that Fran *is* smart. The contradiction was resolved by contracting "Fran *is not* smart," since the sources for its supports were less credible than the sources for "Fran *is* smart."

Wumpus World

We present a demonstration on how the state-constraint-based revision behavior from [Shapiro and Kandefer, 2005] is generalized by our changes to SNeBR. The command *set-order fluent* says that propositional fluents are strictly less entrenched than non-fluent propositions. The *fluent* order was created specifically to replace the original belief revision behavior of the SNeRE *believe* act. In the version of SNeBR used in [Shapiro and Kandefer, 2005], propositions of the form $andor(<0|1>, 1)(p_1, p_2, \ldots)$ were assumed to be state contraints, while the inner propositions, p_1, p_2, etc., were assumed to be fluents. The fluents were less entrenched than the state constraints. We see that the ordering was heavily syntax-dependent.

In our new version, the determination of which propositions are fluents is made by checking for membership of the predicate symbol of an atomic proposition in a list called `*fluents*`, which is defined by the user to include the predicate symbols of all propositional fluents. So the entrenchment ordering defined here uses metaknowledge about the knowledge base that is not represented in the SNePS knowledge base. The command *br-tie-mode manual* indicates that Algorithm 2 should be used. Note that the *xor* connective [Shapiro, 2010] used below replaces instances of *andor(1,1)(...)* from [Shapiro and Kandefer, 2005]. The command `perform believe(wff)` is identical to the command `wff!`, except that the former causes `wff` to be strictly more entrenched than every other proposition during belief revision. That is, `wff` is guaranteed to be *safe* (unless `wff` is itself a contradiction). So we would be using *prioritized* belief revision.

```
;;; Show origin sets
: expert
;;; Always use automatic belief revision
: br—mode auto
Automatic belief revision will now be automatically selected.
;;; Use algorithm 2
: br—tie—mode manual
The user will be consulted when an entrenchment tie occurs.
;;; Use an entrenchment ordering that favors non—fluents over
;;; fluents
set—order fluent
;;; Establish what kinds of propositions are fluents; specifically,
      that the agent is facing some direction is a fact that may
      change over time.
^(setf *fluents* '(Facing))
;;; The agent is Facing west
Facing(west).
;;; At any given time, the agent is facing either north, south, east,
       or west (asserted with forward inference).
xor{Facing(north),Facing(south),Facing(east), Facing(west)}!
;;; The knowledge base as it stands
list—asserted—wffs
wff8!:  ~Facing(north)  {<der,{wff1,wff5}>}
wff7!:  ~Facing(south)  {<der,{wff1,wff5}>}
wff6!:  ~Facing(east)  {<der,{wff1,wff5}>}
wff5!:  xor{Facing(east),Facing(south),Facing(north), Facing(west)}
      {<hyp,{wff5}>}
wff1!:  Facing(west)  {<hyp,{wff1}>}
;;; Tell the agent to believe it is now facing east.
perform believe(Facing(east))
;;; The resulting knowledge base
list—asserted—wffs
wff10!:  ~Facing(west)  {<ext,{wff4,wff5}>}
wff8!:  ~Facing(north)  {<der,{wff1,wff5}>,<der,{wff4,wff5}>}
wff7!:  ~Facing(south)  {<der,{wff1,wff5}>,<der,{wff4,wff5}>}
```

```
wff5!:  xor{Facing(east),Facing(south),Facing(north),Facing(west)} {<
        hyp,{wff5}>}
wff4!:  Facing(east)  {<hyp,{wff4}>}
```

There are three propositions in the no-good when revision is performed: `Facing(west)`, `Facing,east`, and `xor(1,1){Facing(...}`. `Facing(east)` is not considered for removal since it was prioritized by the believe action. The state-constraint `xor(1,1){Facing...}` remains in the knowledge base at the end, because it is more entrenched than `Facing(west)`, a propositional fluent, which is ultimately removed.

6 Conclusions

Our modified version of SNeBR provides decision procedures for belief revision in SNePS. By providing a single resulting knowledge base, these procedures essentially perform maxichoice revision for SNePS. Using a well preorder, belief revision can be performed completely automatically. Given a total preorder, it may be necessary to consult the user in order to simulate a well preorder. The simulated well preorder need only be partially specified; it is only necessary to query the user when multiple beliefs are minimally-epistemically-entrenched within a no-good, and even then only in the case where no other belief in the no-good is already being removed. In any event, the epistemic ordering itself is *user-supplied*. Our algorithm for revision given a well preorder uses asymptotically less time and space than the other algorithm, which uses a total preorder. Our work generalize previous belief revision techniques employed in SNePS.

Acknowledgments

We would like to thank Prof. William Rapaport for providing editorial review, and Prof. Russ Miller for his advice concerning the analysis portion of this paper.

References

[Alchourron and Makinson, 1985] C.E. Alchourron and D. Makinson. On the logic of theory change: Safe contraction. *Studia Logica*, (44):405–422, 1985.

[Alchourron *et al.*, 1985] C. E. Alchourron, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 20:510–530, 1985.

[de Kleer, 1986] J. de Kleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[Gärdenfors and Rott, 1995] P. Gärdenfors and H. Rott. Belief revision. In Gabbay, Hogger, and Robinson, editors, *Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 35–131. Clarendon Press, Oxford, 1995.

[Gärdenfors, 1982] P. Gärdenfors. Rules for rational changes of belief. In T. Pauli, editor, *Philosophical Essays Dedicated to Lennart Åqvist on His Fiftieth Birthday*, number 34 in Philosophical Studies, pages 88–101, Uppsala, Sweden, 1982. The Philosophical Society and the Department of Philosophy, University at Uppsala.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. The MIT Press, Cambridge, Massachusetts, 1988.

[Gärdenfors, 1989] P. Gärdenfors. The dynamics of belief systems: Foundations vs. coherence. *Revue Internationale de Philosophie*, 1989.

[Hansson and Olsson, 1999] S. O. Hansson and E. J. Olsson. Providing foundations for coherentism. *Erkenntnis*, 51(2–3):243–265, 1999.

[Hansson, 1994] S. O. Hansson. Kernel contraction. *The Journal of Symbolic Logic*, 59(3):845–859, 1994.

[Hansson, 1997] S. O. Hansson. Semi-revision. *Journal of Applied Non-Classical Logics*, 7(2):151–175, 1997.

[Hansson, 1999] S. O. Hansson. A survey of non-prioritized belief revision. *Erkenntnis*, 50:413–427, 1999.

[Johnson and Shapiro, 1999] F. L. Johnson and S. C. Shapiro. Says Who? - Incorporating Source Credibility Issues into Belief Revision. Technical Report 99-08, Department of Computer Science and Engineering, SUNY Buffalo, Buffalo, NY, 1999.

[Lakemeyer, 1991] Lakemeyer. On the relation between explicit and implicit beliefs. In *Proc. KR-1991*, pages 368–375. Morgan Kaufmann, 1991.

[Martins and Shapiro, 1988] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35(1):25–79, 1988.

[Shapiro and Johnson, 2000] S. C. Shapiro and F. L. Johnson. Automatic belief revision in SNePS. In C. Baral and M. Truszczynski, editors, *Proc. NMR-2000*, 2000. unpaginated, 5 pages.

[Shapiro and Kandefer, 2005] S. C. Shapiro and M. Kandefer. A SNePS Approach to the Wumpus World Agent or Cassie Meets the Wumpus. In L. Morgenstern and M. Pagnucco, editors, *NRAC-2005*, pages 96–103, 2005.

[Shapiro and The SNePS Implementation Group, 2010] Stuart C. Shapiro and The SNePS Implementation Group. *SNePS 2.7.1 USER'S MANUAL*. Department of Computer Science and Engineering, SUNY Buffalo, December 2010.

[Shapiro, 1992] Stuart C. Shapiro. Relevance logic in computer science. Section 83 of A. R. Anderson and N. D. Belnap, Jr. and J. M/ Dunn *et al. Entailment, Volume II*, pages 553–563. Princeton University Press, Princeton, NJ, 1992.

[Shapiro, 2010] S. C. Shapiro. Set-oriented logical connectives: Syntax and semantics. In F. Lin, U. Sattler, and M. Truszczynski, editors, *KR-2010*, pages 593–595. AAAI Press, 2010.

[Williams, 1994] M.-A. Williams. On the logic of theory base change. In C. MacNish, D. Pearce, and L. Pereira, editors, *Logics in Artificial Intelligence*, volume 838 of *Lecture Notes in Computer Science*, pages 86–105. Springer Berlin / Heidelberg, 1994.