Implementing Δ_{ps} (*PS-Merge*) Belief Merging Operator for Belief Revision

Oscar Chávez-Bosquez[1], Pilar Pozos-Parra[1], Jianbing Ma[2]

[1] Universidad Juárez Autónoma de Tabasco, División Académica de Informática y Sistemas,
Cunduacán, Tabasco,
Mexico

[2] Bournemouth University, School of Design, Engineering and Computing,
Bournemouth, UK

{oscar.chavez, pilar.pozos}@ujat.mx, jma@bournemouth.ac.uk

Abstract. Belief merging aims at combining information from multiple sources while belief revision studies strategies for retracting information in order to maintain consistency when the addition of new evidence to a belief base makes it inconsistent. An ordering of the sentences in the belief base is used to determine priorities among sentences so that those with lower priority can be identified and retracted. This ordering can be difficult to generate and maintain. To address this difficulty, in this paper we show how to generate automatically an ordering of the belief base sentences through the implementation of a belief merging operator. We extend the Δ_{ps} (*PS-Merge*) belief merging operator in order to consider constraints, then we use this extension, called Δ_{ps_μ} (Δ_{ps} under constraints), as a strategy for belief revision. We treat new evidence as a constraint and apply the extended merging operator to obtain the revised belief base. We propose several properties of this operator when compared to other two belief revision operators solving four examples described as real-life scenarios. Finally we show a software prototype based on this approach, called Belief Reviser, freely accessible online.

Keywords. Belief revision, belief merging under constraints, knowledge modeling, decision support systems, Δ_{ps} (*PS-Merge* operator).

1 Introduction

Belief change can be characterized in two ways: in terms of postulates, which are the properties that a changing process must follow in order to produce rational results, and in terms of processes, which consists of a set of predefined steps in order to generate the results. The latter process indeed defines an operator that is also expected to satisfy a set of postulates previously defined in the literature [15].

Belief revision is a framework that characterizes the process of belief change in which an agent revises its beliefs when new evidence is received. Revision always considers new evidence as a better belief [11]. Such new evidence is usually represented in the form of a propositional formula which must be preserved after the revision.

Logic-based belief revision has been extensively studied [1, 3, 10, 11, 19, 28]. An agent's beliefs are usually represented as a belief base and with the help of some order-based strategy the new piece of information is integrated to the belief base to reach a new coherent revised belief base.

Classical belief revision always prefers the new information and thus revises the current beliefs to accommodate new evidence. Most studies of belief revision are based on the AGM (Alchourron, Gardenfors & Makinson) postulates [1] which capture this notion of priority and describe the minimal properties a revision process should have. The AGM postulates formulated in the propositional setting in [19], denoted as R_1-R_6 (see section 3.1), characterize the behavior that a revision operator should comply with. For example,

R_1, called the success postulate, captures the priority of new evidence over the belief base, which requires that the revision result of a belief base K by a proposition μ (new information) should always maintain μ being believed.

Another belief change approach, belief merging, studies strategies for combining information contained in a set of, possibly inconsistent, belief bases (profile) obtained from different sources in order to produce a single consistent belief base.

In this case, the better belief is taken as the collective one, i.e. the result of merging the bases, which according with the approach can be a set of formulae or a single formula.

Several merging operators have been defined and characterized in a logical way. Among them, model-based merging operators [22, 25, 26, 27, 36] obtain a belief base from a set of worlds with the help of a distance measure on worlds and an aggregation function over distances. The closest worlds to the input belief profile are the result of the operator. This framework has a good level of generality, due to the variety of distance functions that may be chosen, however, in the literature almost every proposed operator uses the classical Dalal distance [10]. Implementation of Dalal based frameworks faces a big issue: the computation of the models of every base in the profile, which could be very expensive. Interested readers can refer to [16] for approaches to deal with the implementation of such operators.

An alternative distance between possible worlds has been defined in [5]. This distance is based on the notion of *Partial Satisfiability (PS)*. A merging operator based on PS-distance is also proposed in [5]. The PS-operator has been implemented avoiding the computation of the models of the belief profile. Moreover, this operator can deal with inconsistent bases which have no models, which is an advantage over Dalal based operators that operate only over the profile's models and hence cannot handle inconsistent bases.

In the literature, we found that belief merging can be extended to support integrity constraints, which are typically represented as a set of formulae or a single formula that must be respected by the merging result[1]. In [5], however, the support of integrity constraints is not considered by the proposed operator. Note that as showed in [22], a model-based merging operator can be naturally extended to support integrity constraints. That is, to make the result of the merging fulfil a formula representing the integrity constraints[2] μ, we only need to restrain the search of worlds to the ones that satisfy μ and choose the closest to the input belief profile.

In a previous work [34], we proposed an extension of the Δ_{ps} (*PS-Merge*) belief merging operator and provided a preliminary analysis of its properties. In this paper we *(i)* further elaborate its characterization, *(ii)* discuss the properties of our extension, called Δ_{ps_μ} (Δ_{ps} under constraints), *(iii)* describe corresponding test examples in practical scenarios, and *(iv)* develop a software prototype, called Belief Reviser, to implementing our proposed operators.

We made a comparison of Δ_{ps_μ} against two belief revision operators: Dalal and BHQ operators. Dalal operator is a well known, model-based strategy for belief revision, thus, it is a semantic operator. On the other hand, BHQ operator is a less well-known approach but, unlike Dalal operator, it is a syntax-based strategy based on prime implicates. Namely, it is a syntactic operator. In addition, it can deal with inconsistent belief bases, whilst Dalal operator cannot. This pair of operators are based on different strategies but satisfy the basic postulates, so it is worth a comparison.

This paper is organized as follows. In section 2 we give some definitions and notations on logic and knowledge bases. In section 3 we introduce belief revision and we define BHQ and Dalal operators. In section 4 we introduce belief merging. We describe partial satisfiability and define Δ_{ps_μ}. In section 5 we propose revision through merging via concrete test scenarios which are also described in colloquial language; we compare Δ_{ps_μ} against

---

[1]Here we can see that the integrity constraints being respected in belief merging is similar to the new evidence being respected in belief revision, which has motivated this paper.

[2]For convenience we use the same symbol μ for representing the new evidence in belief revision and the integrity constraints in belief merging contexts.

the other two operators and discuss the results. In section 6 we describe the software prototype developed. Finally in section 7 we give some conclusions and discuss future work.

2 Preliminaries

We consider a language $\mathcal{L}$ of propositional logic using a finite set of atoms or variables $P := \{p_1, p_2, \ldots, p_n\}$. The logic connectives $\rho = \{\neg, \wedge, \vee\}$ are defined as usual.

A Well Formed Formula (WFF), also called formula or sentence, is a propositional formula ψ containing a subset of variables from P and a subset of operators from ρ. Parentheses are used in the traditional way [13]. We can define a formula as follows:

— Each propositional variable is a formula.

— If ψ is a formula, then $\neg \psi$ is a formula.

— If ψ and ϕ are formulae, then $\psi \wedge \phi$ and $\psi \vee \phi$ are formulae.

Unnecessary parentheses may be eliminated in the traditional way.

A *belief base* or *belief theory* K is a finite set of formulae of $\mathcal{L}$ representing the beliefs from a source. In this paper we sometimes identify K with the conjunction of its elements, for example if $K = \{\psi_1, \psi_2, \ldots, \psi_n\}$ then $K = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n$

A literal l is an atom or the negation of an atom. A term D is a conjunction of literals such that $D = l_1 \wedge \cdots \wedge l_k$, with $l_i = p_j$ or $l_i = \neg p_j$. Every formula $\psi \in \mathcal{L}(P)$ can be transformed into a Disjunctive Normal Form (DNF$_\psi$), such that $\psi \equiv$ DNF$_\psi$, where DNF$_\psi$ is a disjunction of terms such that DNF$_\psi$ $=D_1 \vee \cdots \vee D_m$.

A term D is an implicant of a formula ψ iff $D \models \psi$, and it is a prime implicant iff for all implicants D' of ψ, we have $D' \models D$. The set of all the prime implicants of a formula ψ is denoted by $\Xi(\psi)$.

Similarly, a clause C is a disjunction of literals $C = l_1 \vee \cdots \vee l_k$, with $l_i = p_j$ or $l_i = \neg p_j$. Every formula $\psi \in \mathcal{L}(P)$ can be transformed into a Conjunctive Normal Form (CNF$_\psi$), such that $\psi \equiv$ CNF$_\psi$, where CNF$_\psi$ is a conjunction of clauses such that CNF$_\psi$ $=C_1 \wedge \cdots \wedge C_m$.

A clause C is an implicate of a formula ψ iff $\psi \models C$, and it is a prime implicate iff for all implicates C' of ψ, we have $C \models C'$. The set of all the prime implicates of a formula ψ is denoted by $\Pi(\psi)$.

The set of worlds or models of the language is denoted by $\mathcal{W}$, its elements will be denoted by vectors of the form $(w(p_1), \ldots, w(p_n))$, where $w(p_i) = 1$ or $w(p_i) = 0$ for $i = 1, \ldots, n$ and the set of models of a formula ψ is denoted by $mod(\psi)$. K is satisfiable (consistent) iff there is a model of K. If ψ is a propositional formula or a set of propositional formulae then $\mathcal{P}(\psi)$ denotes the set of atoms appearing in ψ. $|\mathcal{P}|$ denotes the cardinality of set $\mathcal{P}$.

A belief *profile* $E = \{K_1, \ldots, K_m\}$ is a multiset (bag) of m belief bases.

Let $\leq_\psi$ a pre-order relation over worlds; $x =_\psi y$ is a notation for $x \leq_\psi y$ and $y \leq_\psi x$, and $x <_\psi y$ is a notation for $x \leq_\psi y$ and $y \not\leq_\psi x$.

3 Belief Revision

Belief revision is a central topic in knowledge representation and reasoning. Belief revision consists in incorporating a new belief, changing as few as possible the original beliefs while preserving consistency [9].

3.1 Belief Revision Characterization

In [1] eight postulates have been proposed to characterize the process of belief revision, which are known as the AGM postulates. Assuming a propositional setting, in [19] this characterization is rephrased yielding the following R_1-R_6 postulates, where K, K_1 and K_2 are belief theories to be revised and μ, μ_1 and μ_2 are pieces of new evidence and $\circ$ is a belief revision operator:

R_1. $K \circ \mu$ implies μ.

R_2. If $K \wedge \mu$ is satisfiable, then $K \circ \mu \equiv K \wedge \mu$.

R_3. If μ is satisfiable, then $K \circ \mu$ is also satisfiable.

R_4. If $K_1 \equiv K_2$ and $\mu_1 \equiv \mu_2$, then $K_1 \circ \mu_1 \equiv K_2 \circ \mu_2$.

R_5. $(K \circ \mu_1) \wedge \mu_2$ implies $K \circ (\mu_1 \wedge \mu_2)$.

R_6. If $(K \circ \mu_1) \wedge \mu_2$ is satisfiable, then $K \circ (\mu_1 \wedge \mu_2)$ implies $(K \circ \mu_1) \wedge \mu_2$.

[33] introduced the notion of *faithful assignment* and provided a representation theorem which shows an equivalence between the six postulates and a revision strategy based on total pre-orders, the formal definitions are as follows [18]:

Definition 1 *Let $\mathcal{W}$ be the set of all worlds (interpretations) of a propositional language $\mathcal{L}$. A function that maps each sentence ψ in $\mathcal{L}$ to a total pre-order $\leq_\psi$ on worlds $\mathcal{W}$ is called a faithful assignment if and only if:*

1. *$w_1, w_2 \models \psi$ only if $w_1 =_\psi w_2$;*

2. *$w_1 \models \psi$ and $w_2 \not\models \psi$ only if $w_1 <_\psi w_2$; and*

3. *$\psi \equiv \phi$ only if $\leq_\psi = \leq_\phi$.*

Theorem 1 (Representation Theorem)
A revision operator $\circ$ satisfies postulates R_1-R_6, iff there exists a faithful assignment that maps each sentence ψ into a total pre-order $\leq_\psi$ such that:

$$mod(\psi \circ \mu) = \min(mod(\mu), \leq_\psi).$$

A belief revision operator can be seen as a function where inputs are a set of beliefs K and a formula μ, and outputs a new set of revised beliefs $K \circ \mu$. In the literature, a number of concrete belief revision operators have been proposed, deploying either semantic or syntactic approaches.

3.2 Dalal Operator

The most prominent belief revision operator, Dalal operator [10] performs a model-based revision measuring minimal change by the cardinality of model change, i.e. it uses a "distance" metric which operates over two worlds w_1, w_2 by counting the number of atoms whose values differ from each other, known as the Hamming distance $dist(w_1, w_2)$. Then this distance helps to define a distance between a belief base K and any possible world w as follows:

$$dist(w, K) = \min_{w' \in mod(K)} dist(w, w').$$

Then a total pre-order $\leq_K$ over possible worlds can be defined as: $w_1 \leq_K w_2$ iff

$$dist(w_1, K) \leq dist(w_2, K).$$

Definition 2 *The revision operator $\circ_{Dalal}$ is defined as:*

$$mod(K \circ_{Dalal} \mu) = \min(mod(\mu), \leq_K). \quad (1)$$

3.3 BHQ Operator

BHQ operator is a prime implicate-based belief revision operator, formula-based yet syntax-insensitive, and do not rely on background information [3]. The operator is defined by first compiling the belief base in its set of prime implicates and then applying the revision operator.

Definition 3 *Let K be a belief base and μ be a formula, $\Pi(K)$ the set of prime implicates of K and $K \perp \mu$ the set of maximal subsets of K consistent with $\neg \mu$. Then the prime implicate-based revision operator, written $\circ_{BHQ}$, is defined as follows:*

$$K \circ_{BHQ} \mu = \mu \wedge \bigvee (\Pi(K) \perp \neg \mu). \quad (2)$$

The operator conjoins the input μ and the disjunction of the maximal subsets of $\Pi(K)$ consistent with μ. Instead of dealing directly with the formulae in the belief base, it deals with the prime implicates of the belief base.

4 Belief Merging

Belief merging aims at combining several pieces of (possibly inconsistent) information coming from different sources [24]. The goal is to produce a single consistent set of information, trying to keep the most of the information.

Belief merging addresses important issues in artificial intelligence and databases, as its applications are many and diverse [4]. For example, a belief merging operator in a multiagent system may reach an agreement on the inputs of a group of agents based on the contradictory beliefs of each member in the group. When agents have conflicting beliefs about the "true" state of the

world, belief merging can be used to determine the "true" state of the world for the group [33].

Several merging operators have been defined and characterized in a logical way. Among them, Δ_{ps} (*PS-Merge*) is a versatile operator which can be used to solve real-world problems, with the advantage of having an efficient implementation [31].

4.1 Partial Satisfiability Merging

Without loss of generality the operator considers only normalized languages so that each belief base is taken as the disjunctive normal form (DNF) of the conjunction of its elements or the conjunctive normal form (CNF) of the disjunction of its elements.

In [5] the notion of *Partial Satisfiability* (PS) is introduced which is a generalization of satisfiability, in which the valuation function $w : \mathcal{L} \to \{0, 1\}$ is extended to $w_{ps} : \mathcal{L} \to [0, 1]$, i.e. the range in a PS-valuation could be any number between 0 and 1. Instead of indicating satisfaction with a Boolean value, partial satisfaction yields a number representing the degree of satisfaction of a formula. If a formula is unsatisfied, its partial satisfiability is 0. If a formula is satisfiable completely (Boolean satisfiable), its partial satisfiability is 1. The partial satisfiability of any other case is between these two values. The authors have proposed two definitions of partial satisfiability: one considers only formulae in DNF and the other, called normal partial satisfiability, considers both forms DNF and CNF. Even when there is a small difference in the valuation of conjunctions, in this paper we consider only normal partial satisfiability, so when we refer partial satisfiability we refer the case of normal partial satisfiability. The difference in the valuation of conjunctions considers a degree of satisfiability when the formula K is unsatisfied and does not contain all the atoms of the language ($|\mathcal{P}(K)| < |\mathcal{P}|$), i.e. when the agent is not satisfied at all in its own beliefs, the partial satisfiability considers a small degree of satisfaction for the atoms not appearing in the formula representing its beliefs [31].

Definition 4 (Normal Partial Satisfiability) *Let* $K \in \mathcal{L}(P)$ *in DNF or CNF,* $w \in \mathcal{W}$, *the Normal Partial Satisfiability of* K *for* w, *denoted as* $w_{ps}(K)$, *is defined as follows:*

— *If* K *is a literal*
$$w_{ps}(K) = w(K),$$

— *If* $K := D_1 \vee \cdots \vee D_n$
$$w_{ps}(K) = max\{w_{ps}(D_1), \ldots, w_{ps}(D_n)\},$$

— *If* $K := C_1 \wedge \cdots \wedge C_n$
$$w_{ps}(K) = \sum_{i=1}^{n} \frac{w_{ps}(C_i)}{n}.$$

Example 1 *The Partial Satisfiability of the belief base* $K = (\neg a \wedge \neg c) \vee (b \wedge \neg c)$ *given* $P = \{a, b, c\}$ *and* $w = (1, 1, 1)$ *is*
$$w_{ps}(K) = max\left\{\frac{w(\neg a) + w(\neg c)}{2}, \frac{w(b) + w(\neg c)}{2}\right\} = \frac{1}{2}.$$

Example 2 *Notice that we can find the Partial Satisfiability of inconsistent belief bases such as* $K = a \wedge (\neg a \vee b) \wedge \neg b$, *given* $P = \{a, b\}$ *and* $w = (1, 0)$, *it is*
$$w_{ps}(K) = \frac{w(a) + max(w(\neg a), w(b)) + w(\neg b)}{3} = \frac{2}{3}.$$

This intuitively means that two out of three conjuncts are satisfied.

4.2 Δ_{ps} Operator

In classical model-based belief merging, the process of merging a profile $\Delta(E)$ defines three distances: a distance from a world to another one $dist(w_1, w_2)$, a distance from a world to a belief base $dist(w, K)$ based on $dist(w_1, w_2)$ and a distance from a world to a profile $dist(w, E)$ based on $dist(w, K)$. The latter distance allows us to define a pre-order between possible worlds with regards to the profile ($\leq_E$). The closest worlds to the profile are the models of the merging result. Partial satisfiability merging is quite similar. The only difference is that we define a distance from a world w to a base K directly without the help of distances between worlds. More precisely, we first define:

$$dist(w, K) = w_{ps}(K),$$

then a distance from a world w to a profile E is defined as follows:

$$dist(w, E) = \sum_{K \in E} dist(w, K),$$

i.e. the distance from a world to a profile is the sum over all the partial satisfiability of every belief base. Finally, a pre-order between possible worlds w.r.t. a profile E is defined as follows:

$$w_1 \leq_E w_2 \quad \text{iff} \quad dist(w_1, E) \geq dist(w_2, E).$$

The models of the merging result are the worlds that are closest to the profile.

$$mod(\Delta_{ps}(E)) = \min(\mathcal{W}, \leq_E). \tag{3}$$

4.3 Δ_{ps} under Constraints

The merging process can be extended straightforwardly when an integrity constraint μ is imposed. To assure that the result of the merging will satisfy the integrity constraints, we can restrict the search to the constraint's models $mod(\mu)$ as follows:

$$mod(\Delta_{ps_\mu}(E)) = \min(mod(\mu), \leq_E). \tag{4}$$

We suppose that the integrity constraint is self-consistent. Moreover, without loss of generality we only consider constraints represented by a single formula μ. If there are n constraints μ_1, μ_2, ..., μ_n we will represent them by the conjunction of the constraints, i.e. we shall consider only the belief merging case under one constraint $\mu = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_n$.

Example 3 (Konieczny & Pino-Pérez, 2002 [23])
At a meeting of co-owners of a block of flats, the chairman proposes for the coming year the construction of a swimming-pool, a tennis-court and a private-car-park. But if two of these three items are built, the rent will increase significantly. We will denote by s, t and p the construction of the swimming-pool, the tennis-court and the private car-park respectively and i will denote the increase of the rent. Two co-owners want to build the three items, and they do not care about the rent increase, $K_1 = K_2 = s \wedge t \wedge p$, the third

thinks that building any item will cause at some time an increase of the rent and wants to pay the lowest rent so he is opposed to any construction, so $K_3 = \neg s \wedge \neg t \wedge \neg p \wedge \neg i$ and finally the last one thinks that the flat really needs a tennis-court and a private car-park but does not want a rent increase i.e. $K_4 = t \wedge p \wedge \neg i$.

The chairman outlines that building two or more items will increase the rent significantly, this fact can not be ignored and worlds in which this fact is falsified must be ignored. These kind of facts are known as integrity constraints.

Let μ be the set of the integrity constraints of the example, it is represented by the single formula $((s \wedge t) \vee (s \wedge p) \vee (t \wedge p)) \to i$. If we consider P the ordered set $\{s, t, p, i\}$ then the worlds $(1, 1, 1, 0)$, $(1, 1, 0, 0)$, $(1, 0, 1, 0)$ and $(0, 1, 1, 0)$ can not be considered as a possible $\Delta_{ps}(E)$ model since these worlds falsify the integrity constraint. It is enough to calculate the Partial-Satisfiability of the bases for worlds in $mod(\mu)$.

The answer given by Δ_{ps} of Example 3 (see table 1) is the world $(1, 1, 1, 1)$, i.e. the decision that satisfies the majority of the group is to build the three items no matter if the rent increases. This decision is also the one obtained using the integrity constraint majority merging operator based on the Σ aggregation function in [23].

5 Revision through Merging

Now, for adapting PS-based framework to a belief revision context it is enough to consider revision as a particular case of merging under constraints, where the profile E is a singleton[3] $E = K$ and constraints μ represents the new information. Thus, revising a belief base K by the new information μ can be defined through merging as follows:

Definition 5 *Let K be a knowledge base and μ be a formula, then the PS-Merge revision of K by μ is defined as:*

$$mod(K \circ_{ps} \mu) = mod(\Delta_{ps_\mu}(K)). \tag{5}$$

---

[3]If the profile is a singleton, we will omit brackets $E = \{K\} = K$.

Table 1. Merging of Example 3 with Δ_{ps_μ}

w	$w_{ps}(K_1)$	$w_{ps}(K_2)$	$w_{ps}(K_3)$	$w_{ps}(K_4)$	$\displaystyle\sum_{i=1}^{4} w_{ps}(K_i)$	
$(1,1,1,1)$	1	1	0	$2/3$	$8/3$	$*$
$(1,1,1,0)†$	1	1	$1/4$	1	$13/4$	
$(1,1,0,1)$	$2/3$	$2/3$	$1/4$	$1/3$	$23/12$	
$(1,1,0,0)†$	$2/3$	$2/3$	$1/2$	$2/3$	$15/6$	
$(1,0,1,1)$	$2/3$	$2/3$	$1/4$	$1/3$	$23/12$	
$(1,0,1,0)†$	$2/3$	$2/3$	$1/2$	$2/3$	$15/6$	
$(1,0,0,1)$	$1/3$	$1/3$	$1/2$	$1/8$	$31/24$	
$(1,0,0,0)$	$1/3$	$1/8$	$3/4$	$1/3$	$37/24$	
$(0,1,1,1)$	$2/3$	$2/3$	$1/4$	$2/3$	$27/12$	
$(0,1,1,0)†$	$2/3$	$2/3$	$1/2$	1	$17/6$	
$(0,1,0,1)$	$1/3$	$1/3$	$1/2$	$1/3$	$3/2$	
$(0,1,0,0)$	$1/3$	$1/3$	$3/4$	$2/3$	$25/12$	
$(0,0,1,1)$	$1/3$	$1/3$	$1/2$	$1/3$	$3/2$	
$(0,0,1,0)$	$1/3$	$1/3$	$3/4$	$2/3$	$25/12$	
$(0,0,0,1)$	$1/8$	$1/8$	$3/4$	$1/8$	$9/8$	
$(0,0,0,0)$	$1/8$	$1/8$	1	$1/3$	$19/12$	

($*$) Indicates the best partial satisfiability of the group, (†) integrity constraint violation

Given the profile is a singleton, the sum of partial satisfiability of the belief base is not necessary, i.e. $\sum_{K' \in E} dist(w, K') = dist(w, K)$, so we can redefine the pre-order in terms of distance from a world to a base as follows:

$$w_1 \leq_K w_2 \quad \textit{iff} \quad dist(w_1, K) \geq dist(w_2, K), \quad (6)$$

which in terms of implementation will reduce the complexity.

5.1 Satisfaction of Postulates

We show now that the operator satisfies, under certain restrictions, postulates R_1-R_6. To prove that we use the representation theorem showed in Theorem 1.

Note that in the settings in Definition 1 and Theorem 1, we only need to consider $\leq_\psi$ instead of considering $\leq_K$ for a general belief base K. Clearly the $\leq_\psi$ is a total pre-order on the possible worlds given that Equation (6) is defined based on a relation over real numbers.

To show that the mapping from ψ to $\leq_\psi$ is faithful, we need to prove the three conditions required in Definition 1. For the first condition, i.e. $w_1, w_2 \models \psi$ only if $w_1 =_\psi w_2$, suppose that $w_1, w_2 \models \psi$ holds, then the partial satisfiability values of ψ for both w_1 and w_2 are 1, i.e. $dist(w_1, \psi) = 1$ and $dist(w_2, \psi) = 1$ which means $w_1 =_\psi w_2$.

For the second condition suppose that $w_1 \models \psi$ and $w_2 \not\models \psi$ hold, then the partial satisfiability of ψ for w_1 is 1 and the partial satisfiability of ψ for w_2 is less that 1 by the definition of partial satisfiability, i.e. $dist(w_1, \psi) > dist(w_2, \psi)$, which means that $w_1 <_\psi w_2$.

However, we must note that the third condition does not hold in general because partial satisfiability is syntax sensitive while condition 3 imposes syntax independence. Nevertheless, for ease of computation, formulae are usually compiled into the disjunction of all its prime implicants (a particular case of DNF) or the conjunction of all its prime implicates (a particular case of CNF) for practical implementation. In this setting we can prove the third condition.

Indeed, in the literature prime implicants decomposition is often used to decide whether two formulae are equivalent or not. The main property satisfied by prime implicants of ψ (denoted by

$\Xi(\psi)$) and prime implicates of ψ (denoted by $\Pi(\psi)$) is shown as follows [30]:

Proposition 1 $\phi \equiv \psi$ *iff* $\Xi(\phi) = \Xi(\psi)$ *iff* $\Pi(\phi) = \Pi(\psi)$.

The representation in prime implicants is unique in the sense that, given a set P of propositional symbols, every proposition built up with symbols of P has exactly one representation in prime implicants that represents a whole family of congruent propositions [30], a similar result can be found using conjunctions of prime implicates.

Therefore, if $\phi \equiv \psi$, let ϕ' and ψ' be the the disjunction of prime implicants of ψ and ψ, respectively, then due to the uniqueness of the prime forms (up to the order of the terms and of the literals that occur within them), we should have ϕ' and ψ' are essentially the same, hence we have $\leq_{\phi'} = \leq_{\psi'}$ because by definition the order of the terms and literals does not affect the partial satisfiability values. Similar results hold for prime implicates.

Given the above analysis, we have the following theorem:

Theorem 2 *If every formula considered have been compiled into the disjunction of all its prime implicants or the conjunction of all its prime implicates, then $\circ_{ps}$ satisfies R_1-R_6.*

While the approach requires compilation to CNF or DNF, it can deal with inconsistent belief bases, which is a significant issue rarely addressed in the literature [21]. Moreover, compilation to DNF or CNF is a problem widely studied and well implemented, so compilation is not a bottleneck in the whole problem. Interested readers can refer to [17, 20] for the compilation problem from a formula to its prime implicants.

5.2 Test Scenarios Adapted to Revision

In order to validate the $\circ_{ps}$ operator and demonstrate some advantages over other operators, we use the following four examples shown in the frames below. The first two examples were borrowed from [3]. The third example shows how our operator can deal with contradictory bases that rarely can any belief merging operator deal with them [21]. The fourth example is taken from [30].

Scenario 1 (Bienvenu, Herzig & Qi, 2008) [3]. A person is dead. An initial review indicates the death is caused by asphyxiation or by suffocation. A latest review indicates the death is caused by asphyxiation or by cardiopulmonary arrest.
$$K = \{a \vee b, \ a \vee c\}$$
$a =$ death by asphyxiation.
$b =$ death by suffocation.
$c =$ death by cardiopulmonary arrest.

Then, a forensic analysis excludes the possibilities of death by asphyxiation and death by suffocation.
$$\mu = \neg a \wedge \neg b$$

Scenario 2 (Bienvenu, Herzig & Qi, 2008) [3]. A family wants to go on vacation and they are planning their journey. The child wants to swim in a beach or taking a flight during the journey. The mother will not go to a colonial city unless she goes shopping in the journey. The father will not go to a beach unless he goes to a colonial city.
$$K = \{a \vee c, \ \neg b \vee d, \ \neg a \vee b\}$$
$a =$ Beach.
$b =$ Colonial city.
$c =$ Flight.
$d =$ Shopping.

After checking the listed preferences and revising the budget, the family finds that neither taking a flight nor shopping is affordable.
$$\mu = \neg c \wedge \neg d$$

We have taken examples from the literature to make a fair comparison and not to bias the results,

also, given the interesting results of BHQ operator which differ from results of the classical Dalal operator.

For ease of understanding, we translate these examples, originally in propositional notations, to real-life situations/scenarios expressed in natural language.

Scenario 3 In an accident, a witness claims that the car starts normally. But the owner argues that the car cannot start or the car has been turned off. Later, the owner's partner remembers that the car is not turned off.

$$K = \{a, \neg a \vee b, \neg b\}$$

a = Car starts.
b = Car turned off.

A technical revision by a mechanic indicates that the car has been turned off.

$$\mu = b$$

Scenario 4 (Marchi, Bittencourt & Perrussel, 2010) [30]. A patient visits a nutritionist for diet advice, and exposes his tastes/dietary habits in the breakfast for a typical day: no watermelon and citrics at all, or no watermelon and no plain water and milk, or no citrics and milk.

$$K = \{(\neg p_3 \wedge \neg p_2) \vee (\neg p_3 \wedge \neg p_1 \wedge p_4) \vee (\neg p_2 \wedge p_4)\}$$

p_1 = Plain water.
p_2 = Citrics.
p_3 = Watermelon.
p_4 = Milk.

The nutritionist then issues his recommendation: no milk and watermelon, or plain water and citrics.

$$\mu = (\neg p_4 \wedge p_3) \vee (p_1 \wedge p_2)$$

Results of $\circ_{ps}$ are compared with existing techniques such as $\circ_{BHQ}$ [3] and the well known Dalal [10] approach.

Table 2 shows the results of each operator. We will expose the use of the three belief revision operators in scenario 1. They apply similarly in the other three scenarios.

Example 4 *The BHQ operator applied to scenario 1 involves the computation of prime implicantes of K, say $\Pi(K)$. In this case $\Pi(K) = K$. Then, the maximal subsets of $\Pi(K)$ consistent with μ are $\{\{a \vee c\}\}$. Finally, $K \circ_{BHQ} \mu$ is $\neg a \wedge \neg b \wedge (a \vee c) \equiv \neg a \wedge \neg b \wedge c$.*

Example 5 *The Dalal's operator returns the set of worlds with minimum distance of the models of μ with respect to K. So $K \circ_{Dalal} \mu$ from scenario 1 is $\{(\neg a, \neg b, c), (\neg a, \neg b, \neg c)\}$ despite the models of K, So $K \circ_{Dalal} \mu$ is $\neg a \wedge \neg b \equiv \mu$.*

Example 6 *$\circ_{ps}$ over scenario 1 considers computing the models of μ: $(0,0,1)$ and $(0,0,0)$, on the assumption that atoms are ordered alphabetically. We can easily verify that the model of $K \circ_{ps} \mu$ is $(0,0,1)$, as shown in Table 3. Then, $K \circ_{ps} \mu$ is $\neg a \wedge \neg b \wedge c$.*

In scenario 1, Dalal's revision result $\neg a \wedge \neg b$ loses information concerning c. Instead, results of BHQ and PS operators $\neg a \wedge \neg b \wedge c$ preserve information on c. This result means that $K \circ_{BHQ} \mu$ and $K \circ_{ps} \mu$ conclude no death by asphyxiation nor death by suffocation but death by cardiopulmonary arrest.

In scenario 2, BHQ revision result is the same as the constraint μ, instead the result of the Dalal operator is $(a \leftrightarrow b) \wedge \neg c \wedge \neg d$, which considers an equivalence between a and b. Result of PS-operator $(a \vee \neg b) \wedge \neg c \wedge \neg d$ considers a disjunction between a and $\neg b$ which assures satisfaction of two conjunctions of input $K = (a \vee c) \wedge (\neg b \vee d) \wedge (\neg a \vee b)$, i.e. we assure that two members of the family are satisfy on their holidays. The advice of $K \circ_{ps} \mu$ is that the family should travel to a beach or not to a colonial city, and avoid taking a flight and shopping. Dalal results also assure the satisfaction of two members but give priority to the father.

Scenario 3 considers the revision of an inconsistent belief base, so Dalal's revision does not apply in this case. Results of BHQ and PS operators retract $\neg b$ in order to produce the consistent belief base $a \wedge (\neg a \vee b) \wedge b \equiv a \wedge b$. In this case both operators result in both the car starts normally and the car has been turned off.

Scenario 4 is an interesting case since it is a more complex example in terms of number of

Table 2. Comparison of approaches

Scenario	K and μ	$K \circ_{BHQ} \mu$	$K \circ_{Dalal} \mu$	$K \circ_{ps} \mu$
1	$K = \{a \vee b,\ a \vee c\}$ $\mu = \neg a \wedge \neg b$	$\neg a \wedge \neg b \wedge c$	$\neg a \wedge \neg b$	$\neg a \wedge \neg b \wedge c$
2	$K = \{a \vee c,\ \neg b \vee d,$ $\neg a \vee b\}$ $\mu = \neg c \wedge \neg d$	$\neg c \wedge \neg d$	$(a \leftrightarrow b) \wedge \neg c \wedge \neg d$	$(a \vee \neg b) \wedge \neg c \wedge \neg d$
3	$K = \{a,\ \neg a \vee b,\ \neg b\}$ $\mu = b$	$a \wedge b$	N/A	$a \wedge b$
4	$K = \{(\neg p_3 \wedge \neg p_2) \vee$ $(\neg p_3 \wedge \neg p_1 \wedge p_4) \vee$ $(\neg p_2 \wedge p_4)\}$ $\mu = (\neg p_4 \wedge p_3) \vee (p_1 \wedge p_2)$	$(\neg p_1 \wedge \neg p_2) \vee (\neg p_3 \wedge p_4)$	$(\neg p_2 \wedge p_3 \wedge \neg p_4) \vee$ $(p_1 \wedge p_2 \wedge p_4) \vee$ $(p_1 \wedge p_2 \wedge \neg p_3)$	$p_1 \wedge p_2 \wedge \neg p_3 \wedge p_4$

Table 3. $K \circ_{ps} \mu$

$w \in mod(\mu)$	$dist(w, K)$	
$(0,0,1)$	$1/2$	*
$(0,0,0)$	0	

(*) Indicates better values

atoms and formulae. Dalal's approach yields a complex recommendation intake for the patient $(p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg p_4) \vee (\neg p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg p_4) \vee (p_1 \wedge p_2 \wedge p_3 \wedge p_4) \vee (p_1 \wedge p_2 \wedge \neg p_3 \wedge p_4) \vee (p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4) \equiv (\neg p_2 \wedge p_3 \wedge \neg p_4) \vee (p_1 \wedge p_2 \wedge p_4) \vee (p_1 \wedge p_2 \wedge \neg p_3)$ Although simplied, it is still a complex formula. $\circ_{BHQ}$ results in $(\neg p_1 \wedge \neg p_2) \vee (\neg p_3 \wedge p_4)$, a simpler breakfast menu, but PS approach results in the most elegant solution: $p_1 \wedge p_2 \wedge \neg p_3 \wedge p_4$: plain water and citrics and no watermelon and milk.

6 Δ_{ps_μ} Implementation

6.1 Algorithm Δ_{ps_μ}

We modified the algorithms proposed in [6, 32] to consider constraints and then formalized the Algorithm 1 mentioned in [34]. The inputs of the algorithm are shown below:

— V : Number of variables of profile E,

— B : Number of bases of E,

— C : Vector of number of conjuncts of each base in E,

— L : Matrix of occurrences of literals by each conjunct of each base in E,

— M : Number of conjuncts of constraints μ,

— L_M : Matrix of occurrences of literals by each conjunct of constraints μ.

Example 7 *Consider the base of scenario 1, with $E = \{K\} = \{(a \vee b) \wedge (a \vee c)\}$, and $\mu = \neg a \wedge \neg b$ then the input data accepted by Algorithm 1 are as follows:*

— $V = 3$

— $B = 1$

— $C = (2)$

— $L = \begin{pmatrix} a & \neg a & b & \neg b & c & \neg c \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

— $M = 2$

— $L_M = \begin{pmatrix} a & \neg a & b & \neg b & c & \neg c \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Complexity of Algorithm 1 is polynomial in V, B and M inputs.

Algorithm 1: Δ_{ps_μ}

Data:
  V : Number of variables of E
  B : Number of bases of E
  C : Vector of number of conjuncts of each base in E
  L : Matrix of occurrences of literals by each conjunct of each base of E
  M : Number of conjuncts of constraints μ
  L_M : Matrix of occurrences of literals by each conjunct of constraints μ

Result:
  $Solution\_Set$: The set of models in $\Delta_{ps_\mu}(E)$

begin
  $Solution \leftarrow \emptyset$;
  $Max\text{-}Sum \leftarrow 0$;
  $W \leftarrow$ Matrix whose rows are the models of μ;
  for $s = 1 \ldots B$ **do**
    $IC_s \leftarrow \sum_{k=1}^{s-1} C_k + 1$;

  for $i = 1 \ldots rows(W)$ **do**
    $Sum \leftarrow 0$;
    for $s = 1 \ldots B$ **do**
      ps-conjunct $\leftarrow 0$;
      for $c = IC_s \ldots IC_s + C_s$ **do**
        $satisfied \leftarrow 0$;
        for $j = 1 \ldots V$ **do**
          if $W_{i,j} = 1$ **then**
            $satisfied \leftarrow L_{c,2j-1}$;
          if $W_{i,j} = 0$ **then**
            $satisfied \leftarrow L_{c,2j}$;
        ps-conjunct $\leftarrow$ *ps-conjunct* $+ satisfied$;

      $PS \leftarrow \dfrac{ps - conjunct}{C_s}$;
      $Sum \leftarrow Sum + PS$;
    if $Sum > MaxSum$ **then**
      $Solution \leftarrow \{i\}$;
      $MaxSum \leftarrow Sum$;
    else if $Sum = MaxSum$ **then**
      $Solution \leftarrow Solution \cup \{i\}$;
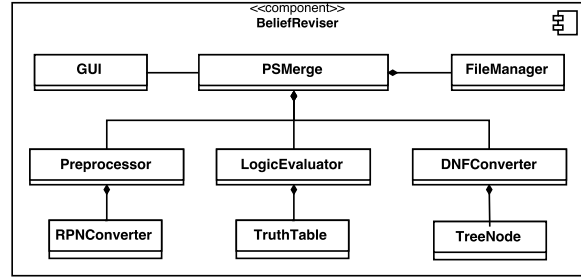 $Solution\_Set = \{i\text{th-row of } W | i \in Solution\}$;

Fig. 1. Belief Reviser UML diagram

6.2 Software Prototype

For users' convenience, we have also developed a software prototype implementing Δ_{ps_μ}. We call it Belief Reviser prototype. The prototype considers the operators in ρ and considers the two missing classical operators $\rightarrow$ and $\leftrightarrow$, i.e $\rho = \rho \cup \{\rightarrow, \leftrightarrow\}$.

Algorithm 1 is implemented in language M using GNU Octave 3.6.2, as it is an open alternative to Matlab® with similar characteristics and performance [14]. We based on existing source code[4], and we design a User Interface (UI) based on the Belief Merger prototype[5] described in [7].

Figure 1 outlines the UML diagram of the Belief Reviser where main classes and its relations are depicted. The prototype was developed under the Open Java Development Kit (OpenJDK) 7 since it is a relatively easy to learn development platform and its object-oriented design. It has multithreaded and platform-independent capabilities, and it is active and robust. Moreover, the platform is free [37].

`PSMerge` class handles all the revision process, instantiating all classes involved while verifying the correct configuration of the running environment. The classes shown in Figure 1 perform different functionalities of the prototype.

---

[4] http://www.utm.mx/~vero0304/PSMerge/PSMerge.htm
[5] http://www.sourceforge.net/p/beliefmerger

6.2.1 Data Preprocessing

`Preprocessor` class parses the belief base and the constraints to build a pair of lists with formulae in Reverse Polish Notation (RPN) notation (postfix notation). Postfix notation, while less common than infix notation in written mathematics, is more explicit in the sense that the operands belonging to a particular operation can be expressed without the need for brackets or operator precedence [12]. Class `RPNConverter` implements Algorithm 2 which shows the implementation of the Dijkstra's Shunting-yard algorithm [41] applied to logical operators in ρ. Operator precedence is shown in Table 4.

Algorithm 2: Shunting-yard algorithm adapted for logical operators

---

Function: `convert` (wff): rpn wff
begin
    rpn = $\emptyset$;
    foreach $c \in$ *wff* **do**
        if $c \in \mathcal{P}(K)$ **then**
          | rpn += c
        else if $c \equiv$ *'('* **then**
          | push(*'('*)
        else if $c \in \rho$ **then**
          while *there is an operator p of higher or equal precedence than c at the top of the stack* **do**
            | pop(p);
            | rpn += p
          push(c)
        else if $c \equiv$ *')'* **then**
          repeat
            | pop(p);
            | **if** $p \in \rho$ **then**
              | rpn += p
          until $p \equiv$ *'('*;
    while $\neg$ emptyStack() **do**
        pop(p);
        rpn += p
    return *rpn*

6.2.2 Logic Formulae Evaluation

`LogicEvaluator` class performs the evaluation of logic formulae in RPN notation with the help of

Table 4. Operator precedence

Priority	Operator	Symbol	Associativity
1	Negation	~	Right to left
2	Conjunction	^	Left to right
3	Disjunction	v	Left to right
4	Implication	>	Left to right
5	Bi-implication	=	Left to right

`TruthTable` class, which contains the matrix of size $n \times m$ where $n = |\mathcal{P}(K)|$ and $m = 2^n$. Algorithm 3 sets up the truth table as follows: column under the first variable should alternate 2^{n-1} *true*'s with with 2^{n-1} *false*'s; column under the second variable should alternate 2^{n-2} *true*'s with with 2^{n-2} *false*'s; continuing until we reach the last column variable.

Algorithm 3: Truth table set up

---

i = 0;
tt[n][m] = 0;
for i=0; i< n; i++ **do**
    j = 0;
    for a=0; a<2^i; a++ **do**
        for c=0; c< $\frac{m}{2^{i+1}}$; c++ **do**
          | tt[i][j++] = 1;
        for c=0; c< $\frac{m}{2^{i+1}}$; c++ **do**
          | tt[i][j++] = 0;
return tt;

The explicitness of the RPN formulae obtained by Algorithm 4 allows us to implement the algorithm for evaluating postfix notation in a straightforward manner. Algorithm 4 shows the RPN evaluator developed using a stack as a helper data structure.

6.2.3 DNF Conversion

`DNFConverter` class performs the process of conversion in two steps. In the first step, an algorithm parses the formula and creates a binary tree according to the operator's precedence (Table 4). In the second step, another algorithm converts the formula to DNF by building the DNF binary tree while walking through the formula binary tree built in the first step. Each node of the tree is an

---

Algorithm 4: RPN evaluator

Function: `evaluate` (interpretation[i], rpn): Boolean
for $c \in rpn$ **do**
    if $c \in \mathcal{P}(K)$ **then**
        | `push` (interpretacion[c])
    else if $c \in \rho$ **then**
        if $c \equiv \neg$ **then**
            | `pop` (b);
            | `push` (¬b)
        else
            | `pop` (b_1);
            | `pop` (b_2);
            | b_3 = `operation`(b_1,b_2,c);
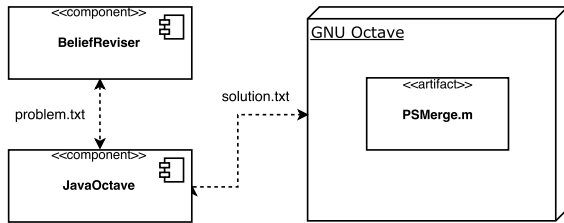 | `push` (b_3)

return `pop()`;

Fig. 2. Belief Reviser component diagram

instance of `TreeNode` class. The conversion of a formula to its DNF is based on the JavaScript code from the Theorem Proving in Propositional Logic (WFFs) web site [2].

6.2.4 PS-Merge Interface

`FileManager` class is the interface between the Java code and the language M implementation of Δ_{ps_μ}. We use JavaOctave 0.6.4 bridging from Java to Octave to run Δ_{ps_μ} in PSMerge.m file. As Δ_{ps_μ} implementation requires an input file with the precise structure (see subsection 6.1), this class builds the input file (problem.txt) and runs Δ_{ps_μ} and the solution is obtained via an output file (solution.txt). Figure 2 illustrates this process.

6.2.5 User Interface (UI)

Figure 3 shows the main elements of the prototype UI. A belief base K is introduced into the top left input box. Constraint μ is introduced into the bottom left input box. Variables are represented by the alphabetical letters (except the 'v' letter), with uppercase and lowercase letters being treated differently. Logical operators are represented by the symbols shown in Table 4. Each belief is separated by commas. An output panel is presented on the right side of the UI where results and debug information are displayed. When needed, formulae is converted to DNF as required by Δ_{ps}.

All the scenarios presented in this paper were solved in this prototype, loaded via text files for each scenario. For example, file format of scenario 1 shown in Figure 3 consists of a first line representing K and a second line representing μ:

```
a v b, a v c
~a ^ ~b
```

Average case performance running the four scenarios is 0.3 seconds. Table 5 shows the test cases executed in the Belief Reviser running on an Alienware M17x laptop with Intel Core i7@2GHz processor, 8 GB RAM, Ubuntu Linux 14.10 OS.

First four test cases correspond to the scenarios, respectively, considering a minimum number of variables. The following four test cases are generated within the prototype and consider a larger number of variables which are unlikely to be solved manually. The performance results of these tests indicate that they can be handled in a reasonable computational time with common hardware.

Table 5. Test cases

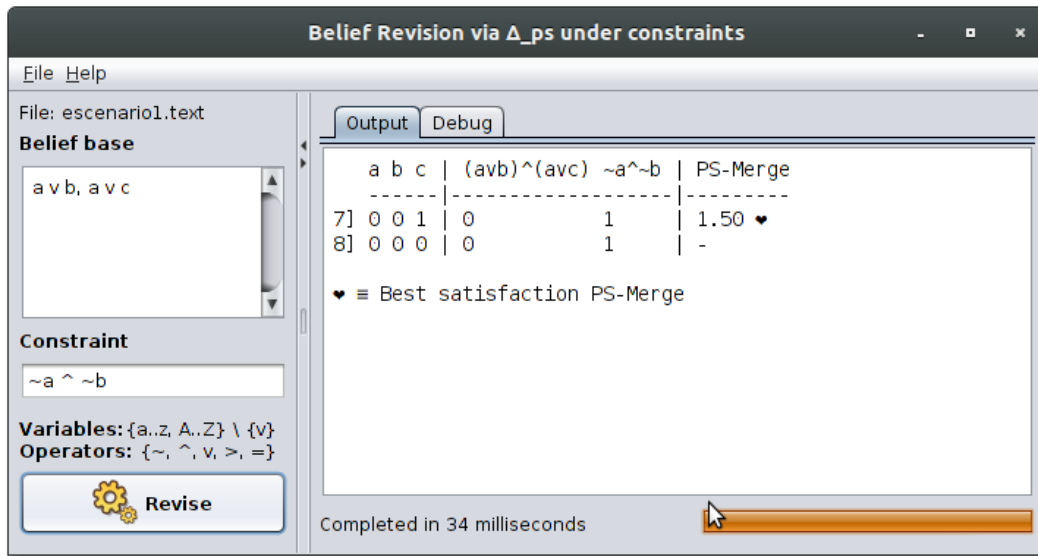| Variables	Time (aprox.)
Small test cases	
3 (Scenario 1)	0.37 s
4 (Scenario 2)	0.34 s
2 (Scenario 3)	0.35 s
4 (Scenario 4)	0.34 s
Medium test cases	
8	9 s
10	50 s
12	1 m 20 s
15	13 m 10 s

Fig. 3. Belief Reviser prototype

Time employed by the prototype is much lower than the time (in minutes) that a human being would spend in solving these scenarios, assuming that the user is familiarized with belief revision concepts. Moreover, examples with greater number of variables and size of K may be unmanageable and error-prone for human beings.

7 Conclusion

Classical belief revision always results in trusting the new evidence. Some approaches need extra information, such as priorities between formulae, in order to process the revision. However, in many cases this extra information is not available. We propose a new revision method considering "flat" belief bases without extra information.

We adapt belief merging in order to carry out belief revision by using belief merging under constraints, which always results in holding the constraints. Instead of creating a new belief revision operator, we consider revision as a particular case of belief merging under constraints where only one a priori base is to be merged.

We have taken into account of the revision postulates that must be satisfied and proved that

under certain syntactical restriction our approach satisfies the AGM postulates. Also, the PS approach is elegant and simple in computation.

We present a comparison of our proposal, Δ_{ps} under constraints, Δ_{ps_μ}, against the well-known Dalal operator, which is a model-based operator, and the BHQ operator, which is formula-based and considers inconsistent formulae; describing in natural language a set of examples with different characteristics. Results are explained and interpreted demonstrating the potential of the PS approach. The three operators shown present dissimilar features, and $\circ_{ps}$ was proved to be easier to implement and more appropriate to some scenarios. For example, $\circ_{ps}$ in scenario 1 retains properly the information of c, in scenario 2 shows to be fair given no preference to the father desires, for scenario 3 handles an inconsistent K and for scenario 4 provides a clear and unambiguous nutritional advice.

It is worth noticing that many revision approaches in the literature use models of operations and hence cannot apply to situations involving an inconsistent belief base. However, PS-based revision can deal with the inconsistency (see scenario 3).

A Belief Reviser software prototype was developed. It automates the process of belief revision given that it is a user-friendly interface implementing Δ_{ps_μ}. We tested the prototype with a set of test cases in different sizes, namely small and medium test cases. Results indicate that problems of medium sizes can be handled by the prototype. The Belief Reviser is free software available online at `http://www.sourceforge.net/p/beliefreviser`.

Future work considers a deep analysis of PS definition in order to propose a new definition that avoids the restriction concerning prime implicant forms, since formulae expressed in prime implicant form satisfy the principle of independence of the syntax. Also, we plan to compare the PS operator against classical operators [38, 40] as well as against more recent proposals [30, 29].

problems, from frameworks for identifying quality knowledge in the Web [39] to model multi-agent systems [35]. Belief revision is one of the most important concepts in studying human reasoning. mathematics are the basis that hold for many other sciences, we believe that many problems could be solved using logic, and belief revision is a promissory technique with a number of applications to real life scenarios.

Finally, we propose to model real-life scenarios, taking examples from different areas such as decision-making from strategic planning [7], or menu planning from nutrition sciences [8]. By this, real scenarios will be validated by experts of each field and comparison of operators will be more interesting and valuable.

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*, 50:510–530.

2. **Allison, L. (2015).** *Theorem Proving in Propositional Logic (WFFs).* `http://www.allisons.org/ll/Logic/Propositional/Wff/`

3. **Bienvenu, M., Herzig, A., & Qi, G. (2008).** Prime Implicate-based Belief Revision Operators. *Proceedings of ECAI '08, IOS Press*, pp. 741–742.

4. **Bloch, I. & Hunter, A. (2001).** Fusion: General concepts and characteristics. *International Journal of Intelligent Systems*, 10(16):1107–1134.

5. **Borja-Macías, V. & Pozos-Parra, P. (2007).** Model-Based Belief Merging without Distance Measures. *Proceedings of AAMAS '07*, pp. 613–615.

6. **Borja-Macías, V. & Pozos-Parra, P. (2009).** Implementing PS-Merge Operator. *Proceedings of MICAI '09*, pp. 39–50.

7. **Chávez-Bosquez, O. & Pozos-Parra, P. (2014).** Modelando la toma de decisiones mediante fusión de creencias. *Recta*, 15(2):141–158.

8. **Chávez-Bosquez, O., Marchi, J., & Pozos-Parra, P. (2014).** Nutritional Menu Planning: A Hybrid Approach and Preliminary Tests. *Research in Computing Science*, (82):93–104.

9. **Creignou, N., Papini, O., Pichler, R., & Woltran, S. (2014).** Belief revision within fragments of propositional logic. *Journal of Computer and System Sciences*, 80(2):427–449.

10. **Dalal, M. (1988).** Investigations into a theory of knowledge base revision: Preliminary report. *Proceedings of the NCAI, AAAI Press*, pp. 475–479.

11. **Darwiche, A. & Pearl, J. (1997).** On the Logic of Iterated Belief Revision. *Artificial Intelligence* 89:1–29.

12. **Drozdek, A. (2004).** *Data Structures and Algorithms in Java*. 2nd ed, Cengage Learning, USA.

13. **Enderton, H. B. (2001).** *A Mathematical Introduction to Logic*. Academic Press, 2nd ed., USA.

14. **Eaton, J.W. (2012).** GNU Octave and reproducible research. *Journal of Process Control*, 22(8):1433–1438.

15. **Gärdenfors, P. (2003).** *Belief revision*. Cambridge University Press, UK.

16. **Gorogiannis, N. & Hunter, A. (2008).** Implementing semantic merging operators using binary decision diagrams. *International Journal of Approximate Reasoning* 49(1):234–251.

17. **Jackson, P. (1990).** Computing prime implicants. *Proceedings of ICAD, LNAI*, 449:543–557.

18. **Katsuno, H. & Mendelzon, A. (1989).** A Unified View of Propositional Knowledge Base Updates. *Proceedings of IJCAI*, pp. 1413–1419.

19. **Katsuno, H. & Mendelzon, A. (1992).** Propositional Knowledge Base Revision and Minimal Change. *Artificial Intelligence* 52:263–294.

20. **Kean, A. & Tsiknis, G. (1990).** An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation*, 9:185–206.

21. **Konieczny, S., Lang, J., & Marquis, P. (2004).** DA^2 merging operators. *Artificial Intelligence* 157(1-2):49–79.

22. **Konieczny, S. & Pino-Pérez, R. (1998).** On the Logic of Merging. *Proceedings of KR '98, Morgan Kaufmann*, pp. 488–498.

23. **Konieczny, S. & Pino-Pérez, R. (2002).** Merging information under constraints: a logical framework. *Journal of Logic and Computation*, 12(5):773–808.

24. **Konieczny, S. & Pino-Pérez, R. (2011).** Logic based merging. *Journal of Philosophical Logic*, 40(2):239–270.

25. **Liberatore, P. & Schaerf, M. (1998).** Arbitration (or how to merge knowledge bases). *Proceedings of IEEE TKDE*, 10:76–90.

26. **Lin, J. & Mendelzon, A. (1999).** Knowledge base merging by majority. *Dynamic Worlds: From the Frame Problem to Knowledge Management*, Kluwer Academic Publishing, pp. 195–218.

27. **Ma, J., Liu, W., & Hunter, A. (2009).** The non-Archimedean polynomials and merging of stratified knowledge bases. *Proceedings of ECSQARU'09*, pp. 408–420.

28. **Ma, J., Liu, W., & Benferhat, S. (2010).** A Belief Revision Framework for Revising Epistemic States with Partial Epistemic States. *International Journal of Approximate Reasoning*, 59:20–40.

29. **Ma, J. & Liu, W. (2011).** A framework for managing uncertain inputs: An axiomization of rewarding. *International Journal of Approximate Reasoning*, 52(7):917–934.

30. **Marchi, J., Bittencourt, G., & Perrussel, L. (2010).** Prime forms and minimal change in propositional belief bases. *Annals of Mathematics and Artificial Intelligence*, 59:1–45.

31. **Borja-Macías, V. & Pozos-Parra, P. (2009).** Implementing PS-Merge Operator. *Proceedings of MICAI'09*, Lecture Notes in Computer Science, Vol. 5845, pp. 39–50.

32. **Pozos-Parra, P., Perrussel. L., & Thévenin, J.M. (2011).** Belief Merging Using Normal Forms. *Proceedings of MICAI'11*, Lecture Notes in Computer Science, Vol. 7094, pp. 40–51.

33. **Pozos-Parra, P., Liu, W., & Laurent, P. (2013).** Dalal's revision without Hamming distance. *Proceedings of MICAI'13*, Lecture Notes in Artificial Intelligence, Vol. 8265, pp. 41–53.

34. **Pozos-Parra, P., Chávez-Bosquez, O., & Ma, J. (2013).** Revision through Belief Merging under Constraints. *Proceedings of AIKED'13*, Recent Advances in Knowledge Engineering and Systems Science, pp. 116–122.

35. **Resconi, G. & Kovalerchuk, B. (2009).** Agents' model of uncertainty. *Knowledge and Information Systems* 18(2): 213–229.

36. **Revesz, P.Z. (1997).** On the Semantics of Arbitration. *Journal of Algebra and Computation*, 7(2):133–160.

37. **Russell J. & Cohn, R. (2012).** *OpenJDK*. 1st ed., USA.

38. **Satoh, K. (1988).** Nonmonotonic reasoning by minimal belief revision. *Proceedings FGCS '88*, pp. 455–462.

39. **Su, K., Huang, H., Wu, X., & Zhang, S. (2006).** A logical framework for identifying quality knowledge from different data sources. *Decision Support Systems*, 42(3):1673–1683.

40. **Weber, A. (1986).** Updating propositional formulas. *CEDS '86*, pp. 487–500.

41. **Wilson, R. (1988).** *An introduction to dynamic data structures*. 1st ed., McGraw-Hill; USA.