Belief Revision in Propositional Knowledge Bases

George Flouris and Dimitris Plexousakis
{fgeo,dp}@csd.uch.gr
Department of Computer Science, University of Crete
P.O.Box 2208, Heraklion, Crete, GR-714 09 GREECE
tel: +30(81)393500, fax: +30(81)393501

Abstract. A most crucial problem in knowledge representation is the revision of knowledge when new, possibly contradictory, information is obtained (belief revision). In this paper, we address this problem when the knowledge base is a set of expressions in propositional logic. We introduce a new representation of propositional expressions using 2-dimensional tables and show why this representation is more expressive than classical propositional logic. We exploit this increased expressiveness to devise a solution to the problem of belief revision in propositional knowledge bases and describe how to represent data and knowledge with tables, as well as a simple method to perform revisions under this new notion. Finally, we provide a theoretical foundation of our method and compare it with proposed algorithms from the literature.

1 Introduction

One of the fundamental problems in knowledge bases (KB) is the revision of knowledge in the face of new, possibly contradictory information. This paper addresses the problem of revising beliefs in KBs consisting of a set of expressions in propositional logic.

The updating methods of knowledge are by no means obvious, even when we are concerned with the intuitive processes only. Let us consider the simple example of knowing a fact A as well as the proposition $A \rightarrow B$. One obvious implication of the above is the fact B (modus ponens), which could be inserted into our KB as a new fact. Let us now consider the negation of B ($\neg B$) entering into the base as a new piece of knowledge (update). This contradicts our assumption that B is true, so we will have to give up some (or all) of our previous beliefs or we would result to an inconsistent KB. Alternatively, we could reject the update as non-valid. Even in this trivial example, it is not clear which approach should be taken. Extra-logical factors should be taken into account, like the source and reliability of each piece of information or some kind of bias towards or against updates.

This and other similar problems have been addressed by several scientists, including philosophers, computer scientists, logicians and others, in an effort to provide us with an intuitively correct method of belief updating. An excellent introductory survey of such efforts by Gärdenfors may be found in [2]. Our paper concentrates on the description of a new method of representing propositional expressions and the effects of this representation on the problem of belief revision.

2 General Properties of Belief Revision Algorithms

Our main goal while designing our belief revision algorithm is the concurrence of the results with the human intuition. Human beings constantly change their beliefs as new information arrives and any rational belief revision algorithm should do the same operation, in a similar

way. One of the fundamental properties of any updating scheme should be the retention of as many as possible of the old data. This property is also known as Dalal's principle of persistence of prior knowledge ([3]).

This is a widely accepted property of belief revision algorithms, in contrast to other considerations, which are heavily debated. One such consideration deals with the representation of the KB. There are two general types of theories concerning this representation: *foundational* and *coherence* theories ([8]). Foundational theorists argue that knowledge should consist of a set of reasons, where only some beliefs (called *foundational*, or *reasons*) can stand by themselves; the rest are derived from the most basic (foundational) beliefs. On the other hand, coherence theorists believe that each piece of knowledge has an independent standing and needs no justification, as long as it does not contradict with other beliefs. The approach chosen greatly influences the algorithms considered, as the foundational approach must take into account the causality relationships, which is not the case for the coherence approach.

A similar consideration is the storage of the KB in the form of a *belief set* where all beliefs must be explicitly stored, or in the form of a *belief base*, where only reasons are stored, enough to reproduce the whole set. For practical applications, we have to work on belief bases instead of belief sets, because the implications of any belief base may be an infinite (or an extremely large) set of propositions. However, this does not exclude the possibility of using the coherence paradigm; we simply produce the part of the belief set needed at run-time (using a theorem prover) and ignore any causality relationships possibly implied by the use of the theorem prover. An additional problem concerning belief bases is the selection of the base, as there may be several belief bases for the same belief set.

The selection of the belief base is related to another important consideration, which is the problem of *iterated revisions*. It can be shown that there are sequences of revisions which give counter-intuitive results if we process each one individually. A solution to this problem is to process the sequence of revisions as a whole ([9, 10]). To deal with iterated revisions, some of our previous assumptions must be revisited. The main problem regarding the one-update algorithms is the fact that the belief base is not properly selected after each update, because the algorithms are only concerned with the result of the update and not with how this result occurred. This can cause the loss of valuable information. The proposed solution is based on the principle that two KBs should be considered equivalent if, in addition to the logical equivalence of the bases themselves, they will give equivalent results to all possible updates as well. This is the basic principle governing the algorithms of iterated belief revision ([9]). It is also pointed out in [10], where the difference between *belief sets* (knowledge only) and *epistemic states* (knowledge including information on how to revise it) is discussed.

To achieve the above goals, our method will use Nebel's proposition ([4]) for the selection of the belief base. Nebel proposes the use of a belief base consisting of propositional sentences representing specific observations, experiments, rules etc out of which our knowledge is derived. This approach follows the foundational paradigm, as the propositions in the base are the foundational beliefs and the rest are implied directly or indirectly by them.

Under most updating schemes, updates are considered more reliable than the old data. This is generally a good practice, as updates are usually regarded as the latest information about the world and can be assumed correct. However, this may not be true in several cases, as the new data may come from a noisy or otherwise unreliable source. In

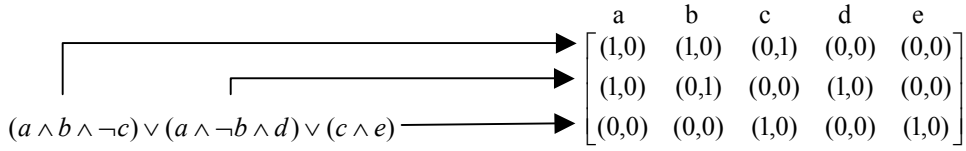
order to overcome the problem, we will assign a non-negative real number to each belief, which will represent its reliability. We will call this number the *Reliability Factor* of the belief and we will use the abbreviation *RF*. When a piece of contradictory information is used to revise our knowledge, at least some of the existing data (or the update itself) must be rejected. The use of the RF implies that the piece(s) of data to be rejected should be the one(s) with the lowest RF. Notice however that there are cases when small changes in existing pieces of data (or the update) are enough to accommodate the update without introducing any inconsistencies. No data rejection is necessary in this case. The selection of the data to reject or change is not an easy one, because there may be several ways to accommodate an update. Finding all possible ways to do so requires evaluating all possible subsets of our belief base and comparing all the ways of removal of the inconsistency in terms of RF cost. This is a computationally expensive operation.

3 Table Transformation

Considerations such as the above, led us to the search of an algorithm that would give us the contradictions, the possible ways of removal and the cost of removal per case at the same time. This can be done by the application of the *table transformation*, which transforms an expression of any finite propositional language into a 2-dimensional table.

The transformation can be applied to any proposition; however, for technical reasons it is better to use the proposition's disjunctive normal form (DNF). Any well-formed formula of propositional logic has an equivalent DNF expression, so this is not a restriction. The transformation returns a 2-dimensional table of ordered pairs of non-negative real numbers.

Each atom of the language is assigned to one column of the table, and there are as many lines in the table as the number of disjuncts in the DNF of the propositional expression. In the figure below, we show the transformation of the propositional expression P=(a∧b∧¬c)∨(a∧¬b∧d)∨(c∧e) into its respective table. We suppose that the language consists of 5 atoms, namely a, b, c, d and e, so the table has 5 columns. The expression is already in DNF and it has 3 disjuncts, so the number of lines in the table is 3. As far as the contents of the table are concerned (called *elements* hereof), the procedure is the following: an element has the value of (1,0) if and only if the respective atom in the respective disjunct appears as a positive atom; if negative the value is (0,1); if the atom does not appear at all, then the value of the element is (0,0). The application of these rules on the expression P results in the table below:



Example of a table transformation

Each element in the table represents the type of membership of one atom in one disjunct of the proposition. This membership may be *positive* (1,0), *negative* (0,1) or *zero* (0,0), depending on how the atom appears in the respective disjunct. Additionally, we could assign weights to atoms. Moreover, elements of the form (x,y) where both x and y are

positive (non-zero), are allowed and are called *contradictory*. A line that contains at least one contradictory element is a *contradiction*. For example, Table A below has weights for each element and it has only one non-contradictory line, the second one.

$$A = \begin{bmatrix} (1,5) & (0.2,1) & (5,0) \\ (2,0) & (0,0) & (1,0) \\ (0,1) & (0,3) & (1,1) \end{bmatrix}$$ (example of a weighted table with contradictions)

Note that a contradictory element indicates the existence of both an atom and its negation in a conjunction, which is a contradiction in propositional logic.

Let us suppose that our knowledge is represented by Table A above. Table A has 3 lines, each one representing a conjunction. This indicates that the world being modeled by A has 3 *possible states*, under our current knowledge. In reality, the world has only one state, but our knowledge is incomplete, so we don't know which is the correct state yet. Therefore, each line in a table represents one possible world. Moreover, each element of the table shows our confidence in each atom (or its negation), per possible world. This will prove very important later on, as new updates may cast doubts on some of our beliefs and we should know which one is more reliable.

4 Formal Definitions

In order to define the above transformation more formally, we will first introduce some notations. The set of all 2-dimensional tables with k lines and n columns whose elements are taken from the set $R^+ \times R^+$ will be denoted by MR(k,n). The union of all such sets for all $k \in N^*$ will be denoted by MR(*,n). We define multiplication of a number with a table and addition of tables in the usual mathematical way. Finally, we define the operation '|' upon two tables $A,B \in MR(*,n)$ as the simple juxtaposition of their lines.

In the rest of this chapter, we will describe the main definitions and theoretical results related to the transformation and its properties. Proofs will be omitted due to lack of space, but they can be found in [11].

Definition 1. We define the following tables, which will be useful at later stages:

- Let $A \in MR(1,n)$ be a table of the form [(0,0) … (0,0) (1,0) (0,0) … (0,0)], where the element (1,0) is in the k-th column. Table A will be called a *k-atom* or a *positive k-atom* and we will use the notation A_k.
- Let $A \in MR(1,n)$ be a table of the form [(0,0) … (0,0) (0,1) (0,0) … (0,0)], where the element (0,1) is in the k-th column. Table A will be called an *inverse k-atom* or a *negative k-atom* and we will use the notation A'_k.
- The table $A=[(0,0) (0,0) … (0,0)] \in MR(1,n)$ is called the *n-true table* and we will use the notation T_n.
- The table $A=[(1,1) (1,1) … (1,1)] \in MR(1,n)$ is called the *n-false table* and we will use the notation F_n.

The informal analysis made in the previous section shows that our goal is to assign the k-atoms to the atoms of our propositional language. T_n will be assigned to the constant T and F_n will be assigned to the constant F.

Definition 2. We call an *interpretation* of the space MR(*,n) any sequence $(a_1,a_2,\ldots,a_n)$, where $a_i \in \{0,1\}$, i=1,2,…,n. The set of all interpretations of MR(*,n) will be denoted by I(n).

The above definition is similar to the definition of interpretations in a propositional language. The transformation of a logical interpretation to a table interpretation (and vice-versa) can be made in a straightforward manner by assigning the logical constant F (falsehood) to number 0 and the logical constant T (truth) to 1.

Definition 3. An element $(x,y) \in R^+ \times R^+$ of a table $A \in MR(*,n)$ is called:

- *positive* iff x>0, y=0
- *negative* iff x=0, y>0
- *zero* iff x=y=0
- *contradictory* iff x>0, y>0.

Definition 4. Let $B \in MR(1,n)$ and $I=(a_1,a_2,\ldots,a_n) \in I(n)$. We say that table B is *satisfied* by the interpretation I iff there exist $b_i \in R^+$, i=1,2,…,n, such that:

$$B = \sum_{i=1}^{n} (a_i \cdot b_i \cdot A_i + (1-a_i) \cdot b_i \cdot A_i') .$$

If $B \in MR(k,n)$, for k>1, then $B=B_1|B_2|\ldots|B_k$, where $B_i \in MR(1,n)$, i=1,2,…,k and we say that B is *satisfied* by I iff there exists $i \in \{1,2,\ldots,k\}$ such that B_i is satisfied by I.
We define the *models* of $B \in MR(*,n)$ (denoted by mod(B)) to be the set of interpretations that satisfy B, ie.
$mod(B)=\{I \in I(n) : B \text{ is satisfied by } I\}$.

Notice that each b_i in the above definition is arbitrarily chosen and represents the RF of the respective atom. This implies that the RF does not affect the models of a table.
Moreover, for any b_i, there are two possibilities:

- $a_i \cdot b_i=0$ and $(1-a_i) \cdot b_i=bi \geq 0$ (iff $a_i=0$)
  or
- $ai \cdot b_i=b_i \geq 0$ and $(1-a_i) \cdot b_i=0$ (iff $a_i=1$)

This remark, along with the definition of satisfiability and k-atoms imply proposition 1 below.

Proposition 1. Let $B=[(x_i,y_i)] \in MR(1,n)$. Then $mod(B)=I_1 \times I_2 \times \ldots \times I_n$, where for each $i \in \{1,2,\ldots,n\}$:

- $I_i=\{0\}$ iff (x_i,y_i): negative
- $I_i=\{1\}$ iff (x_i,y_i): positive
- $I_i=\{0,1\}$ iff (x_i,y_i): zero
- $I_i=\varnothing$ iff (x_i,y_i): contradictory.

The above definition is similar to the definition of satisfiability used in propositional logic. Positive elements indicate the existence of a positive atom, negative elements the existence of a negative atom, zero elements indicate no atom, while contradictory elements indicate the existence of both an atom and its negation.

Definition 5. We define the operation of *disjunction* (denoted by $\vee$) between two tables $A,B \in MR(*,n)$ as: $A \vee B = A|B$.

Definition 6. Let $B \in MR(k,n)$, $C \in MR(m,n)$, such that $B=B_1|B_2|\ldots|B_k$, where $B_i \in MR(1,n)$, $i=1,2,\ldots,k$ and $C=C_1|C_2|\ldots|C_m$, where $C_j \in MR(1,n)$, $j=1,2,\ldots,m$. We define the operation of *conjunction* (denoted by $\wedge$) between B and C as:

$$B \wedge C = \bigvee_{i=1}^{k} \bigvee_{j=1}^{m} (B_i + C_j)$$

Definition 7. Let $B=[(x_i,y_i)] \in MR(1,n)$ and $X=\{i \in \{1,2,\ldots,n\} : x_i \neq 0\}$, $Y=\{i \in \{1,2,\ldots,n\} : y_i \neq 0\}$. Then we define the operation of *negation* (denoted by $\neg$) on B as follows:

- If $X=Y=\varnothing$ then $\neg B = F_n$.
- If $X=\varnothing$, $Y \neq \varnothing$ then $\neg B = \bigvee_{i \in Y} y_i \cdot A_i$.
- If $X \neq \varnothing$, $Y=\varnothing$ then $\neg B = \bigvee_{i \in X} x_i \cdot A_i'$.
- If $X \neq \varnothing$, $Y \neq \varnothing$ then $\neg B = (\bigvee_{i \in X} x_i \cdot A_i') \vee (\bigvee_{i \in Y} y_i \cdot A_i)$.

Let $B \in MR(k,n)$ such that: $B=B_1|B_2|\ldots|B_k$, where $B_i \in MR(1,n)$, $i=1,2,\ldots,k$. We extend the operation of negation in MR(k,n) as follows:

$$\neg B = \bigwedge_{i=1}^{k} (\neg B_i).$$

Proposition 2. For any two tables $A,B \in MR(*,n)$ the following equations hold:

- $mod(A \vee B) = mod(A) \cup mod(B)$
- $mod(A \wedge B) = mod(A) \cap mod(B)$
- $mod(\neg A) = I(n) \setminus mod(A)$.

The above proposition is crucial for the table transformation. It shows that the operations of conjunction, disjunction and negation in tables have the same properties (with respect to interpretations) as the respective operations in propositional logic.

Definition 8. Let $L=\{\vee,\wedge,\neg,(,),T,F,a_1,a_2,\ldots,a_n\}$ be a finite propositional language. We define the *table transformation function* $TT:L^* \rightarrow MR(*,n)$ recursively as follows:
For the constants we define:

$TT(T)=T_n$,
$TT(F)=F_n$.

For any atom $a_i \in L$, we define:

$TT(a_i)=A_i$, for $i=1,2,\ldots,n$.

For any two propositions $A,B \in L^*$ we define:

$TT(A \vee B) = TT(A) \vee TT(B)$
$TT(A \wedge B) = TT(A) \wedge TT(B)$
$TT(\neg A) = \neg TT(A)$.

We also define the *inverse table transformation function* $TTI:MR(*,n) \rightarrow L^*$ as follows:
Let $B=[(x_i,y_i)] \in MR(1,n)$ and $X=\{i \in \{1,2,\ldots,n\} : x_i \neq 0\}$, $Y=\{i \in \{1,2,\ldots,n\} : y_i \neq 0\}$. Then we define TTI on B as follows:

If X=Y=∅ then TTI(B)=T.

If X=∅, Y≠∅ then $TTI(B) = \underset{i\in Y}{\wedge}(\neg a_i)$.

If X≠∅, Y=∅ then $TTI(B) = \underset{i\in X}{\wedge} a_i$.

If X≠∅, Y≠∅ then $TTI(B) = (\underset{i\in X}{\wedge} a_i) \wedge (\underset{i\in Y}{\wedge}(\neg a_i))$.

In general, let B∈MR(k,n) such that: $B=B_1|B_2|\ldots|B_k$, where $B_i \in MR(1,n)$, i=1,2,…,k. In this case we extend the TTI function in MR(k,n) as follows:

$$TTI(B) = \underset{i=1}{\overset{k}{\vee}} TTI(B_i).$$

5 Updates and Queries

Now that we know how to transform any expression into a table, let us see why this is useful for the revision of beliefs. We will assume that both the knowledge and the update are represented by tables. For the moment, we will additionally assume that both the base K and the update M have only one line, so they both represent only one possible world. In this special case, the update will be defined as the *addition* of the two tables, because the inclusion of the update M in our knowledge increases our trust in all the information that M carries, effectively increasing our reliance in each atom and/or its negation. This adjustment may or may not be enough to force us to change our beliefs. Let us see one example:

$$K = \begin{bmatrix}(0,1) & (3,0) & (0,0)\end{bmatrix}, M = \begin{bmatrix}(3,0) & (2,0) & (1,0)\end{bmatrix}$$

Using the TTI function we can easily verify that the tables represent the expressions K=¬A∧B (base) and M=A∧B∧C (update). The tables additionally show the reliability per atom in each proposition. In this case, the negation of atom A (¬A) is believed with an RF of 1 in K, but M should force us to abandon this belief as atom A is believed with an RF of 3 in M. In atom B, there is no contradiction between K and M; however this revision should increase our confidence in the truth of B. Finally, in atom C, we know now that C is true, with a reliance of 1; we had no knowledge regarding C before. The resulting table is:

$$K' = K \bullet M = \begin{bmatrix}(0,1) & (3,0) & (0,0)\end{bmatrix} + \begin{bmatrix}(3,0) & (2,0) & (1,0)\end{bmatrix} = \begin{bmatrix}(3,1) & (5,0) & (1,0)\end{bmatrix}$$

The proposition related (via the TTI function) to the resulting table is: A∧¬A∧B∧C. Note that the resulting table K′ is a contradictory table (and the related proposition is a contradiction too). This is not generally acceptable, as a contradictory KB actually contains no information. The result should have been A∧B∧C, as showed by the previous analysis. We will deal with this problem later (in fact this is not a problem at all!).

In the general case where any of the tables contains more than one line, each line represents one possible world. In order to be sure that the correct world will be represented by a line in the resulting table, we must add each line of table K with each line of table M, creating one line per pair in the resulting table K′. Notice that this is actually the conjunction of the two tables.

In general, the resulting table of a revision may contain contradictory lines, like in the previous example. One may argue that contradictory lines represent contradictory possible worlds, so they contain false information and they could as well be deleted. However, this is not entirely true. A single faulty revision may create a contradiction in an otherwise correct possible world (line). Once a contradiction is introduced, it can by no means be removed; therefore, we must be careful before discarding a contradictory line as

non-true. On the other hand, even if a line (possible world) contains no contradictions at all, this is by no means a guarantee that it is entirely true. It could be too far from the real world; we just don't know it yet. Therefore, our policy is to keep the table as-is, even if some (or all) lines are contradictory.

The solution to the problem of contradictory lines when it comes to answering queries is to transform (some of) them to non-contradictory. Whenever a query is executed on the KB, our answer must be based on the "most correct" lines, ignoring the rest. The criterion for correctness of a line is a quantity named *line reliability (RL)*. We also define the *element reliability (RE)*. The reliability of an element (x,y) should depend on the numbers x, y of the element and the reliability of a line should depend on the reliability of its elements. The selection of the line and element reliability formulas is crucial for the algorithm. Different selections of those formulas actually produce different revision algorithms. We propose an entropy-based measure:

$$RE(x, y) = (x + y) + x \cdot \log(\frac{x}{x + y}) + y \cdot \log(\frac{y}{x + y})$$

$$RL_i = \sum_j RE_{ij} \ .$$

The number of lines to be selected as most correct for the query should be a user-defined parameter depending on the application. For example, we could take the k most reliable ones; or all lines with reliability more than a given number; or use any other method that the user may find suitable.

Subsequently, we define the *respective* or *related logical expression* of a line by a procedure similar to the TTI function. Each element (x,y) of the line is changed into the element (1,0) if x>y, (0,1) if x<y and (0,0) if x=y. Following that, the inverse of the table transformation (TTI) is used to transform the line into a conjunction of atoms (or their negations). The result is the related logical expression of the line. Once the most reliable lines are selected for the query, they are transformed into their respective logical expressions, and the disjunction is taken. The result is the base upon which the query is executed.

Note that the above method does not actually change the table; it temporarily transforms it for the needs of the query. It keeps one table for the base, and not each individual update, but the retention of contradictory lines implies that the individual updates are kept in an encoded fashion. Moreover, iterated revisions are supported, as we keep track of all the previous updates and RFs. Finally, the rejection of the old data when answering queries is made according to the RF and RL, and is minimal with respect to these quantities.

6 Complexity Issues

Regarding the computational complexity of our update algorithm, it is easy to see that for a KB with n_0 lines and an update of n_1 lines, the resulting table will have $n_0 \cdot n_1$ lines. Supposing that the underlying propositional language has m atoms, all tables will have m columns. To calculate each element 2 additions must be performed, which implies that the operations required for the calculation of the resulting table are of $O(n_0 \cdot n_1 \cdot m)$. After k consecutive updates, with n_1, n_2,…, n_k lines respectively per update, the resulting table will have $n_0 \cdot n_1 \cdot \ldots \cdot n_k$ lines. This implies that the number of lines in the table representing the final KB increases exponentially with the number of updates, and so does the computational cost for each new update as well as the memory space required to store the base.

This may be unacceptable in most cases, so one could decide to reject some of the lines of the table by a procedure based on line reliability, which is called *abruption*. The number of lines to be removed should be an application-dependent, user-defined parameter, representing a trade-off between knowledge integrity and processing speed.

7 Comparison with Previous Work

Many different propositions exist for the problem of belief revision. Most of them are based on different assumptions and placed on a different context. Dalal in [3] proposed a very promising method of revising beliefs. He provided an algorithm for updating propositional KBs and formalized the notion of *minimal change*. He proved in [3] that his method generally retained more knowledge than the up-to-then proposed algorithms. It can be shown ([11]) that his revision method can be simulated using our semantics, despite the different assumptions on the design of the algorithm. More specifically, the following theorem holds:

Theorem 1. Let p, q be two satisfiable propositional expressions in DNF and let r be the revision of p with q under Dalal's algorithm ($r=p\bullet^{D}q$). Moreover, let $P\in MR(*,n)$ the table related to p via the TT function, using an RF of 1 for all atoms, $Q\in MR(*,n)$ the table related to q via the TT function, using an RF of 2 for all atoms and $R\in MR(*,n)$ the table resulting by the update of P with Q under our framework ($R=P\bullet Q$). Finally, we define element and line reliability:

$$RE(x, y) = 1 - \min\{x, y\}$$
$$RL_i = \sum_j RE_{ij} .$$

If we select the lines of R with maximum reliability, ie if $R\in MR(k,n)$ we select the lines i such that $RL_i \geq RL_j \ \forall j\in\{1,2,\ldots,k\}$, then the resulting propositional expression (to be used in queries) is logically equivalent to the expression r as defined above.

An alternative approach to the belief revision problem was presented by Alchourron, Gärdenfors and Makinson in a series of papers ([2, 5]), back in 1985. Their idea was to recede from the search of any specific algorithm for a while and attempt to formalize the notion of revision. As a result, a set of widely accepted properties of any belief revision algorithm was introduced, in the form of 6 *basic* and 2 *supplementary* postulates which were in fact a set of logical propositions (named *AGM postulates* after the initials of the authors). By specifying those postulates, a series of important theoretical results could be proved. One such result is that Dalal's algorithm satisfies all 8 postulates ([1]). Therefore, our algorithm, with the parameters specified in Theorem 1, is compatible with the AGM postulates.

8 Conclusions and Future Work

In this paper, we presented an innovative representation of propositional expressions. We successfully applied this representation to the problem of belief revision. The introduction of the Reliability Factor (RF) and the quantitative nature of the table representation introduce an increased expressiveness in propositional logic, allowing the use of features not normally available. This could possibly speed up existing algorithms or provide solutions to existing

problems. We believe that much more work needs to be done in order to fully exploit its capabilities.

The operation of *contraction* ([2, 5]) is an important operation not currently included in our model. It could be defined in the same way as update, with one difference being that instead of adding the lines, we should subtract them. This approach forces us to extend our model to include elements with negative numbers. Negative numbers introduce several difficulties and force us to make some important changes in the theoretical foundation of the table transformation. This model extension can be found in [11].

Moreover, the search for the general conditions (parameters) under which the algorithm satisfies the AGM postulates is an ongoing effort, as well as the simulation of other algorithms, like those proposed in [6] and [7] by Williams, in [4] by Nebel etc.

References

1. Katsuno, Mendelzon, "Propositional Knowledge Base Revision and Minimal Change", KRR-TR-90-3, March 1990, *Technical Reports on Knowledge Representation and Reasoning*, University of Toronto.
2. P. Gärdenfors, "Belief Revision: An introduction", pp. 1-20 in *Belief Revision*, ed. by P. Gärdenfors, Cambridge University Press, 1992.
3. M. Dalal, "Updates in Propositional Databases", Technical Report, DCS-TR-222, Department of Computer Science, Rutgers University, February 1988.
4. Bernhard Nebel, "A Knowledge Level Analysis of Belief Revision", *In Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 301-311, 1989.
5. Makinson David, "How to Give it up: A Survey of some Formal Aspects of the Logic of Theory Change", 1985, *Synthese* 62, pp. 347-363.
6. Mary-Anne Williams, "Anytime Belief Revision", 1997, *International Joint Conference on Artificial Intelligence*, 1997.
7. Mary-Anne Williams, David M. Williams, "A Belief Revision System for the World Wide Web", Web Site: http://u2.newcastle.edu.au/webworld/ai-internet.html.
8. P. Gärdenfors, "The dynamics of knowledge and belief: Foundational vs. coherence theories", *Revue Internationale de Philosophie* 44, pp. 24-46. Reprinted in *Knowledge, Belief and Strategic Interaction*, ed. by C. Bicchieri and M. L. dalla Chiara, Cambridge University Press, Cambridge, 1992, pp. 377-396.
9. Paolo Liberatore, "The Complexity of Iterated Belief Revision", *In Proceedings of the Sixth International Conference on Database Theory (ICDT'97)*, pages 276-290, 1997.
10. A. Darwiche and J. Pearl, "On the Logic of Iterated Belief Revision", UCLA Cognitive Systems Laboratory, Technical Report (R-202) In R. Fagin (Ed.), *Proceedings of the 1994 Conference on Theoretical Aspects of Reasoning about Knowledge (TARK '94)*, Pacific Grove, CA, 5-23, March 13-16, 1994. In *Artificial Intelligence*, 89(1-2):1--29, 1997.
11. Flouris George, Plexousakis Dimitris, "Belief Revision using Table Transformation", Technical Report TR-290, July 2001, ICS-FORTH, forthcoming.