BReLS: A System for the Integration of Knowledge Bases

Paolo Liberatore and **Marco Schaerf**
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198, Roma, Italy
Email: liberato@dis.uniroma1.it schaerf@dis.uniroma1.it

Abstract

The process of integrating knowledge coming from different sources has been widely investigated in the literature. Three distinct conceptual approaches to this problem have been most succesful: belief revision, merging and update.

In this paper we present a framework that integrates these three approaches. In the proposed framework all three operations can be performed. We provide an example that can only be solved by applying more than one single style of knowledge integration and, therefore, cannot be addressed by anyone of the approaches alone.

The framework has been implemented, and the examples shown in this paper (as well as other examples from the belief revision literature) have been successfully tested.

1 Introduction

In this paper we introduce a new framework for the integration of information coming from different sources. The proposed framework allows for the formalization of complex domains where the information can have different degrees of reliability and can arrive at different time points.

Many formalisms for the integration of knowledge bases have been introduced in the literature in the last years. Among the most relevant ones we have belief revision theory, due to Alchourrón, Gärdenfors, and Makinson [1] and update, discussed by Katsuno and Mendelzon [7].

In the original formulation of the belief revision theory, due to Alchourrón, Gärdenfors, and Makinson [1], only a revision operator is considered. Revision has been initially defined as the operation that allows modifying our knowledge referring to a static world (i.e. a scenario that does not change) when some new piece of information comes to be known. The work of Winslett [17] made clear that when information is about a scenario that may change, the expected properties of the integration operator are different. For this reason, the operator of revision has in this case a different name: update. A detailed discussion of the differences between revision and update has been carried on by Katsuno and Mendelzon [6].

Both revision and update assume that the new piece of information is correct. This assumption must be removed if we want to merge two pieces of information, referring the same time point, that have the same degree of reliability. This is a very common scenario: we have information coming from difference sources, with the same degree of reliability, that may contradict each other. The case in which two sources contradict each other is indeed the most interesting one (if all pieces of information can be put together without obtaining an inconsistency, we can simply do it). There is no consensus, in the literature, about the name to give to the operation of combining knowledge bases with the same degree of reliability: Revesz [15] and Liberatore and Schaerf [10, 11] use the name *arbitration*, while Lin and Mendelzon [13] and Lin [12] prefer the term *merging*. Konieczny and Pino Perez [8] show that there is a semantical distinction between arbitration and merging. Namely, they show that there are scenarios in which arbitration is more appropriate, and other ones in which the operation to perform is merging. This issue is relatively new, so there is no agreement on the semantics yet.

Summarizing, the three operators for knowledge integration can be described as follows.

Revision: integration of two pieces of information, in

which one is considered fully reliable, while the other one may be partially incorrect;

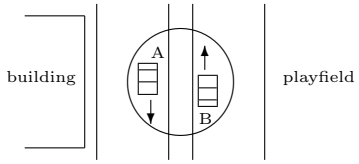
Update: integration of two pieces of information, both fully reliable, referring to two different time point (some changes may have occurred in between);

Merging/Arbitration: integration of two (or more) pieces of information having the same degree of reliability.

The framework we present allows for all of the above forms of reasoning to be performed. We now present an example that calls for the use of both revision and merging.

Example 1 *Let us consider the following scenario: we have a system to control speed and direction of a car. In order to decide what is the current position of the car we have four sources of information: the estimated position of the car using some satellite system; the information coming from sonar sensors mounted on the car (they can percept the presence of buildings along the road); the vision system, which can perform the same operation; and finally, we have the map of the city.*

Suppose that the satellite indicates the position of the car to be inside the circle in the following picture:



As a result, the car may be either in the position marked A, or in the position marked B. In logical terms, this can be represented as $\texttt{pos}_A \not\equiv \texttt{pos}_B$*, i.e. the car is either in position A or in position B (but not both). From the map we know that if the car is in position A, there should be a building on its right side, and if it is in position B, there should be a building on its left side. As a result, we have* $\texttt{pos}_A \rightarrow \neg\texttt{building\_left} \wedge \texttt{building\_right}$*, and* $\texttt{pos}_B \rightarrow \texttt{building\_left} \wedge \neg\texttt{building\_right}$*.*

The sensors say that there is a building on the left, and nothing on the right. This piece of information can be represented as $\texttt{building\_left} \wedge \neg\texttt{building\_right}$*. The information coming from the vision system is that there are buildings both on the left and on the right. In propositional terms, this can be represented as* $\texttt{building\_left} \wedge \texttt{building\_right}$*.*

In this example, there is a pair of sources of information which are fully reliable, that is, the position as given by the satellite and the map of the city. In this case, these pieces of information are consistent with each other, but they are not complete. What we know is that the car is in one of two possible positions. Since we want the exact position of the car, we should use the information coming from the two other sources, which are less reliable.

The problem can be solved by observing that, if the car is in position A, we have to assume that both the sensors and the vision system are faulty. On the other hand, the car being in position B means that only the vision system is wrong. As a result, this second possibility should be preferred.

In this example, we have both revision and merging, as there is a set of sources of information, and some of them have the same degree of reliability. Note that, in more complex scenarios, there may be more than two sources, all having a different degree of reliability (e.g. three sources each having a degree of reliability different from the other ones).

Let now consider an example in which we have more than one time point, but revision is also needed because we have a contradiction that cannot be solved by assuming that something changed.

Example 2 *We consider a client-server architecture. A client computer sends requests of service to the server; the server receives them and executes some kind of computation, and immediately sends a message back to the client informing it that the request has been processed. The server may not execute the requests either because it is too busy (and then the requests are delayed) or it is crashed (in this case, requests are never executed).*

Suppose that, after sending a service request, the client receives no answer. After some time, it assumes that the server is crashed, and thus the request has not been processed. Expressing such a knowledge in the propositional language, we have, at time 1, the following pieces of knowledge: $\neg\texttt{service\_executed}$ *and* $\texttt{crashed}$*.*

Eventually, the client receives an answer from the server, stating that the service has been executed: this also implies that the server is not crashed. Now, let us consider how reliable our knowledge is:

$\neg\texttt{service\_executed}$ at time 1. *Since the client did not receive any acknowledgment of execution, we can believe that the server did not process the request (this is because the server sends the answer*

immediately after processing the request and we assume that the server does not crash after the execution but before sending the answer).

`crashed` at time 1. *This is an assumption made by the client. Indeed, if the answer is late from the server, the client first believe that the server is busy, and only after some time it starts to think that it crashed.*

`service_executed` $\wedge \neg$`crashed` at time 2.
We have received the answer, thus the service has been executed, and the server is not crashed. This piece of information is fully reliable.

Now, we have also some assumptions on change: a server cannot restart working after a crash. It is quite clear what happened in this scenario. At time 1 the server was not crashed: it was simply forced to delay the processing of the request. The processing was executed between time 1 and 2.

In the scenario of the example both revision and update are necessary. Indeed, the initial knowledge (that is, the knowledge at time 1) was that the server was crashed and the service was not executed. We have then a revision of this information, because we discover that the server was not crashed, after all. We also have an update, since the execution of the request happened between time 1 and 2.

In order to help experimenting with our system, we have implemented BReLS. The current implementation is available at the URL
`http://www.dis.uniroma1.it/~liberato/brels`
A CGI interface allows the system to be run over the Internet.

2 Syntax

In this section we describe the syntax of BReLS. As shown in Example 1, different pieces of information may have different degrees of reliability. In BReLS, we write:

$$\texttt{source}(i) : K$$

In words, we mean that the piece of knowledge K has reliability i. Knowledge is expressed using propositional formulas, while reliability is given as positive integer numbers. Thus, K must be a propositional formula and i an integer number such that $i > 0$. Our assumption is that the satellite system and the map always provide fully reliable information, while the sensor and the vision systems may be wrong. This can be formalized by assigning an higher degree of reliability to the information provided by the satellite and inferred from the map. As a result, Example 1 can be encoded as follows:

$$\begin{array}{ll}
\texttt{source}(2): & \texttt{pos}_A \not\equiv \texttt{pos}_B \\
\texttt{source}(2): & (\texttt{pos}_A \rightarrow \neg\texttt{building_left} \wedge \\
& \texttt{building_right}) \\
& \wedge(\texttt{pos}_B \rightarrow \texttt{building_left} \wedge \\
& \neg\texttt{building_right}) \\
\texttt{source}(1): & \texttt{building_left} \wedge \neg\texttt{building_right} \\
\texttt{source}(1): & \texttt{building_left} \wedge \texttt{building_right}
\end{array}$$

In BReLS, information having degree of reliability i is always considered more reliable than any number of pieces of information with reliability $j < i$. In other words, a source saying that a variable is true is always believed, even if there are hundreds of less reliable sources saying that the variable is false.

Let us now introduce time in our framework. Borrowing Sandewall's syntax [16], we write time in square brackets:

$$\texttt{source}(i) : [t]K$$

This formula means that the information K is believed to hold at time t with a degree of reliability i. When time is not specified, we assume by default time point 1. Note that we assume that time is discrete and linear.

The fact that scenarios may be modified is formalized using the idea of *changes*. A change is what in the reasoning about actions field is known as action, and in belief revision as events [3]. Essentially, a change is something that modifies the world of interest. The default assumption is that changes do not happen. However, we can explicitly say that a change may happen with a given "penalty". The general form of change statements is:

$$\texttt{change(i)}: \quad [t]l$$

In this statement, i is a nonnegative integer, while l is a literal. It means that l may become true, and the penalty associated to this event is given by the integer i. The greater the value of the penalty i, more unlikely is the change. In most cases, the penalty of changes is independent from time. In this case, we have the simplified form:

$$\texttt{change(i)}: \quad l$$

which means that the penalty of the change making l true is i for any time point. We have also two ways

for expressing the penalty of changes when this is independent of the literal:

$$\texttt{change(i)}: \quad [t]$$
$$\texttt{change(i)}$$

The first statement denotes that any change at time t has penalty i, while the second one states that any change at any time point has penalty i. When the change statements are inconsistent, we assume that the most specific one holds. For instance, if we have $\{\texttt{change(1)}, \texttt{change(2)} : l\}$ then all changes have penalty 1, except l which has penalty 2. More precisely, the penalty of the change that makes l true between time points t and $t+1$ is specified as follows:

1. If there is a statement $\texttt{change(i)} : [t]l$, the penalty is i, regardless of the other change statements;
2. Else if the knowledge base contains a change statement $\texttt{change(j)} : [t]$, then j is the penalty;
3. Else if there is a change statement $\texttt{change(k)} : l$, then the penalty is k;
4. Else if there exists a change statement $\texttt{change(m)}$, then the penalty is m;
5. Else the penalty is 1;

Let us now consider how Example 2 can be expressed using the syntax of BReLS. The knowledge about time points can be simply expressed as:

$\texttt{source(2)}: \quad [1]\neg\texttt{service\_executed}$
$\texttt{source(1)}: \quad [1]\texttt{server\_crashed}$
$\texttt{source(2)}: \quad [2]\texttt{service\_executed} \wedge \neg\texttt{server\_crashed}$

Indeed, the pieces of knowledge with reliability 2 are those expressing things known for certain. On the other hand, $[1]\texttt{server\_crashed}$ is just an assumption made, thus it is less reliable. Now, any change may happen, except that a server cannot recover from a crash. As a result, we give penalty 1 to any change except that in which the server becomes non-crashed.

$\texttt{change(1)}$
$\texttt{change(4)}: \quad \texttt{server\_crashed}$

The first statement says that all changes have penalty 1 (where not otherwise specified), thus they may happen. The second statement says that the variable `server_crashed` cannot switch from false to true. By default, the statement `change(1)` can be omitted, that is, every change whose penalty is not given is assumed to have penalty 1. A penalty 0 for a change means that there is no problem in assuming that the change happened, that is, we do not make the default assumption that the change does not happen.

3 Semantics

In this section we first introduce the notion of dynamic model, and then we provide two different semantics.

3.1 Models

Suppose the time points we are dealing with are $\{1, \ldots, T\}$, and the propositional variables are $X = \{x_1, \ldots, x_n\}$.

Definition 1 *A* static model *is a truth assignment to the variables in X.*

A static model represents the state of the world in a given time point. In order to express the state and the changes in all considered time points, we introduce the notion of dynamic model.

Definition 2 *A* dynamic model *is composed of two parts:*

1. *an initial (static) model M_0, for the time point 0;*
2. *a sequence of (possibly empty) sets of changes $C_1, \ldots C_n$ for each pair of consecutive time points.*

This is a very general definition, without any explicit definition of what changes are. In BReLS, the only changes considered are those setting the value of a variable. Changes are represented by literals: for instance the change that makes a false is represented by $\neg a$. Given M_0 and the sequence $C_1, \ldots C_n$ we can unambigously define the static model at time t as the result of iteratively applying the changes in $C_1 \ldots C_t$ to the initial model M_0.

3.2 Preference of Static Models.

Let M be a static model, and F a propositional formulas. We define the distance between M and F as:

$$\text{dist}(M, F) = \min(\{\text{hamm}(M, N) | N \in \text{mod}(F)\}, \leq)$$

where $\text{hamm}(M, N)$ is the Hamming distance between the models M and N (the number of atoms to which they assign a different truth value). In words: the distance between a static model M and a propositional

formula F is the minimal Hamming distance between M and any model of F.

Let us consider a knowledge base composed of two formulas with the same reliability and referring to the same time point:

$$\begin{aligned} \texttt{source(1)}: &\quad K_1 \\ \texttt{source(1)}: &\quad K_2 \end{aligned}$$

Since we prefer models that are as close as possible to the formulas in the knowledge base, we define the preference of a static model M as:

$$\text{pref}(M) = \text{dist}(M, K_1) + \text{dist}(M, K_2)$$

The models of a knowledge base are those with the minimal value of the preference function, thus the models in which the sum of the distances is minimal. Let us now consider a knowledge base in which there is another formula with a greater degree of reliability, for instance:

$$\begin{aligned} \texttt{source(1)}: &\quad K_1 \\ \texttt{source(1)}: &\quad K_2 \\ \texttt{source(2)}: &\quad K_3 \end{aligned}$$

Since K_3 is the most reliable formula, all minimal models should satisfy it, thus the preference of a model cannot be defined as a sum of distances. The preference of a model is instead defined as an array having one element for each level of reliability in the knowledge base. In our example, there are two formulas with reliability 1 and one formula with reliability 2, thus we need an array with two elements. The array is defined as:

$$\text{pref}(M)[i] = \sum_{F \text{ has reliability } i} \text{dist}(M, F)$$

In the specific example above, the array has two elements, whose value is:

$$\begin{aligned} \text{pref}(M)[1] &= \text{dist}(M, K_1) + \text{dist}(M, K_2) \\ \text{pref}(M)[2] &= \text{dist}(M, K_3) \end{aligned}$$

The ordering between two models M_1 and M_2 is defined as follows, where P is the maximal level of reliability of formulas in the knowledge base.

$$\begin{aligned} M_1 \prec M_2 \quad \text{iff} \quad & \exists i\ (1 \leq i \leq P) \text{ such that} \\ & (\text{pref}(M_1)[i] < \text{pref}(M_2)[i]) \text{ AND} \\ & \forall j.\ (i < j \leq P) \text{ implies} \\ & (\text{pref}(M_1)[j] = \text{pref}(M_2)[j]) \end{aligned}$$

This definition ensures that if the distance of M_1 to the formulas with the maximal reliability is less than the distance of M_2, then M_1 is preferred, regardless of the formulas having lower reliability. The distance to formulas with reliability $P-1$ only matters when the distance to formulas with reliability P is the same for both models.

As a result, if there is only one formula with the greatest reliability, then all minimal models of the knowledge base have static models satisfying that formula (if consistent). This is what is expressed by the AGM postulate of success: if a formula with the greatest priority is consistent, then all models of the revised knowledge base must satisfy it. On the other hand, our ordering satisfies the principle that if a formula with a lower reliability is consistent with the ones with greatest priority, the result of revision is the logical conjunction of them.

3.3 Preference between Dynamic Models

So far, we have defined the array of preference of a static model with respect to a set of formulas referring to the same time point. We now consider knowledge bases with time. In order to define the preference between dynamic models, we have to take into account two facts: 1. there are formulas about different time points, and 2. there are penalties associated with changes.

Let KB be a knowledge base, and let $KB(t,p)$ be the set of its formulas with time t and priority p. Let $DM(t)$ be the static model associated with time t in the dynamic model DM. The preference array of a dynamic model DM is defined as follows.

$$\text{pref}(DM)[p] = DM_p + \sum_{t=1}^{T} \sum_{F \in KB(t,p)} \text{dist}(DM(t), F)$$

where DM_p is the number of changes with penalty p in the model DM. The definition of $\prec$ is exactly as in the case of static models. When all formulas are fully reliable (that is, when their degree of priority is greater than those of changes), this definition can be rephrased as: consider the minimal set of changes needed to explain the observations. This is similar to the principle of "actions to explain observation", which is studied in reasoning about actions [9, 2, 14].

There are two possible ways for defining the models of a knowledge base. We call these two possible semantics *backward* and *pointwise*. The first one is the simplest to explain, and allows for expressing scenarios in which knowledge at later time may rule out some

initial models. The second one is based on the principle of "pointwiseness": each initial model should be updated separately (this is one of the basic principles of update according to Katsuno and Mendelzon [6]).

Backward Semantics. We define the ordering $\prec$ between dynamic models from the arrays of preference, as we did for static models. The models of a knowledge base are the minimal dynamic models.

$$\mathrm{mod}(KB) = \min(\{DM\}, \prec)$$

Pointwise Semantics. This is the case in which update should be pointwise, that is, there should be a different set of minimal (dynamic) models for each minimal initial (static) model. The set of models of a knowledge base is the set of minimal models according to the ordering $\trianglelefteq$, defined as:

$$DM \trianglelefteq DM' \quad \text{iff} \quad DM(1) = DM'(1) \text{ and } DM \preceq DM'$$

In other words, given the ordering $\trianglelefteq$ above, the models of KB are defined as:

$$\mathrm{mod}(KB) = \min(\{DM\}, \trianglelefteq)$$

This definition can be reformulated as follows:

$$\mathrm{mod}(KB) = \bigcup_{M \text{ minimal initial model}} \min(\{DM \mid DM(1) = M\}, \prec)$$

This way, for each minimal initial (static) model M, we take the set of minimal dynamic models such that $DM(1) = M$. This equivalent formulation shows that the pointwise semantics obeys the principle that each initial model should be updated separately, which is taken by Katsuno and Mendelzon [6] as a basic principle of update.

Logical entailment under these semantics is defined in a classical fashion. More precisely, a knowledge base KB implies a formula $[t]Q$ under the backward (pointwise) semantics, denoted by $KB \models_{\preceq} [t]Q$, if and only if Q is true at time t in any minimal, under the $\preceq$ ($\trianglelefteq$) ordering, model of the knowledge base.

4 Expressing Revision, Update and Merging

In this section, we show how revision, merging, and update can be encoded in our framework. Namely, we show how "classical" revision can be performed using BReLS, and then we show the same for update. We remark, however, that these operators can be considered as *restrictions* of the full semantics. What is really new in BReLS is the fact that they can be performed together on the same knowledge base.

We can express belief revision in BReLS as follows: First, K and P hold at the same time point, and we have more confidence in P. As a result, we have:

$$KB = \left\{ \begin{array}{ll} \texttt{source(1)}: & K, \\ \texttt{source(2)}: & P \end{array} \right\}$$

In BReLS, statements without an explicit time point specification are assumed to refer to time 1.

What in the AGM framework is denoted as $K * P$ is actually the formula representing our knowledge after the integration of K and P. As a result, we can say that the result of revising K with P is given by:

$$\mathrm{mod}(K * P) = \{DM(1) \mid DM \text{ is a model of } KB\}$$

In other words, encoding K and P in BReLS according to the principles of revision, we obtain the revision operator $*$ defined by the above formula. Note that all statements refer to the same time point, hence, there is no difference in using the backward or pointwise semantics. It is quite easy to prove that $*$ is indeed a revision operator satisfying all AGM postulates for revision.

Let us now consider update. The update operator is used when we have two pieces of knowledge referring to two different time points. In the initial formulation of update [17] these two pieces of knowledge are considered to be fully reliable. However, they can contradict each other. In order to explain the contradiction, we assume that something change in the world.

The most evident difference between revision and update is that, when the two involved pieces of knowledge are consistent to each other, the result of revision is always the union (conjunction) of them, while the result of update may be different [6].

Let K be the piece of knowledge referring to the first time point, and P the other one. Assume that the first time point is 1, and the second one is 2. The assumptions we made can be encoded in BReLS as follows:

$$KB = \left\{ \begin{array}{ll} \texttt{source(2)}: & [1]K, \\ \texttt{source(2)}: & [2]P, \\ \texttt{change(1)} & \end{array} \right\}$$

In words, K holds at time 1 while P holds at time 2. Moreover, both pieces of knowledge are reliable, in the sense that models with changes are always preferred

over models that do not satisfy K or P. The result of the update is defined as the formula representing the static models referring to time 2, that is,

$$\begin{aligned}\mathrm{mod}(K \circ P) \quad = \quad & \{DM(2) \mid DM \text{ is a model of } KB \\ & \text{using the pointwise semantics } \}\end{aligned}$$

It can be easily proved that $\circ$ is equivalent to Forbus' update: for any pair of formulas K and P it holds $K \circ P \equiv K \circ_F P$, where $\circ_F$ is the update defined by Forbus [4].

Note, however, that basic update operators like Forbus' one suffer from some drawbacks. For example, they do not allow revision of initial state. In other words, while knowledge about time 1 gives some information about time 2, the converse does not hold: information about time 2 does not extend our knowledge about the previous time point. The litmus test example [3] shows a scenario in which such backward inference should be done.

Let us now consider merging, or arbitration. This is the operation of integrating two knowledge bases having the same degree of reliability [15, 13, 10, 12, 11, 8]. The assumption is that the two knowledge bases refer to the same time point. Let the two knowledge bases be represented by the formulas K_1 and K_2. It is straightforward to express them in BReLS:

$$KB = \left\{ \begin{array}{ll} \texttt{source(1)}: & K_1, \\ \texttt{source(1)}: & K_2 \end{array} \right\}$$

The result of arbitration or merging is the formula representing the result of the integration of the two pieces of information. The arbitration operator defined by encoding K_1 and K_2 as above is thus:

$$\mathrm{mod}(K_1 \triangle K_2) = \{DM(1) \mid DM \text{ is a model of } KB\}$$

Let us now consider the properties of this operator. First of all, it is commutative, in the sense that $K_1 \triangle K_2 \equiv K_2 \triangle K_1$. Second, if K_1 and K_2 are consistent to each other, we have $K_1 \triangle K_2 \equiv K_1 \wedge K_2$. Moreover, $K_1 \triangle K_2$ is always consistent. Finally, $K_1 \triangle K_2$ only depends on the sets of models of K_1 and K_2, that is, $\triangle$ is a syntax-independent operator. Since these are the four basic property of merging, we can conclude that the way BReLS integrates knowledge bases obeys the basic principles of merging.

On the negative side, $\triangle$ does not fulfill the basic property of arbitration. Namely, it does not hold $\mathrm{mod}(K_1 \triangle K_2) \subseteq \mathrm{mod}(K_1) \cup \mathrm{mod}(K_2)$. This means that we can perform merging, but not arbitration. However, this property can be enforced using a slightly different encoding:

$$KB' = \left\{ \begin{array}{ll} \texttt{source(2)}: & K_1 \vee K_2, \\ \texttt{source(1)}: & K_1, \\ \texttt{source(1)}: & K_2 \end{array} \right\}$$

and then defining ∇ like $\triangle$:

$$\mathrm{mod}(K_1 \nabla K_2) = \{DM(1) \mid DM \text{ is a model of } KB'\}$$

Since the ∇ operator has the same four properties of $\triangle$ and it holds $\mathrm{mod}(K_1 \nabla K_2) \subseteq \mathrm{mod}(K_1) \cup \mathrm{mod}(K_2)$, we conclude that ∇ is a valid arbitration operator. We note also that it is trivial to extend both $\triangle$ and ∇ in order to integrate more than two knowledge bases with the same degree of reliability.

5 Remarks and Conclusions

As a final step in our analysis we charcterize the complexity of the two semantics of BReLS. The problem we analyze is the problem of entailment, that is, given a knowledge base KB and a formula $[t]Q$, decide whether KB implies $[t]Q$. The results are:

- Deciding whether $KB \models [t]Q$ holds under the pointwise semantics is Π_2^p-complete.
- Deciding whether $KB \models [t]Q$ under the backward semantics is Δ_2^p-complete.

These are good news, since the complexity of reasoning in BReLS is not higher than the complexity of reasoning in other, simpler, systems for belief revision where priorities and time cannot be expressed.

Summing up our contribution, in this paper we have shown how revision, merging and update can be implemented in a system able to perform all of them on the same knowledge base.

The specific choice of the ordering of models in BReLS is fixed. While we can deal with many example using this semantics, there are examples that cannot be correctly dealt by BReLS. For instance, while we can express a problem similar to the Stolen Car Example [5], we cannot express the problem itself, because there is no way to state that the mileage of a car can increase only if the car is outside the parking lot. More work is needed to fully understand how the semantics can be extended to deal with such scenarios.

Further work is also needed in the analysis of more efficient algorithms. Indeed, in its current implementation, BReLS can deal only with small instances. For

this reason, if we want to really use it in real-world domains, efficient algorithms should be developed and implemented.

Acknowledgements

Part of the work reported in this paper has been done while the first author was visiting Linköping University. He thanks Erik Sandewall for his hospitality and support. This work has been partially supported by the Italian Space Agency (ASI).

References

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] C. Baral, M. Gelfond, and A. Provetti. Representing actions: Laws, observations and hypothesis. *Journal of Logic Programming*, 1996.

[3] C. Boutilier. Generalized update: belief change in dynamic settings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1550–1556, 1995.

[4] K. D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1273–1278, 1989.

[5] N. Friedman and J. Y. Halpern. A knowledge-based framework for belief change, part II: Revision and update. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 190–200, 1994.

[6] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 387–394, 1991.

[7] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[8] S. Konieczny and R. Pino Perez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 488–498, 1998.

[9] R. Li and L. Pereira. What is believed is what is explained. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, pages 550–555, 1996.

[10] P. Liberatore and M. Schaerf. Arbitration: A commutative operator for belief revision. In *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI'95)*, pages 217–228, 1995.

[11] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[12] J. Lin. Integration of Weighted Knowledge Bases. *Artificial Intelligence*, 83(2):363–378, 1996.

[13] J. Lin and A. Mendelzon. Knowledge base merging by majority. Manuscript, 1994.

[14] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 167–179, 1998.

[15] P. Z. Revesz. On the semantics of theory change: Arbitration between old and new information. In *Proceedings of the Twelfth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'93)*, pages 71–82, 1993.

[16] E. Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[17] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.