Merging Ontologies via Kernel Contraction

Raphael Cóbe, Fillipe Resina, Renata Wassermann

[1]Institute of Mathematics and Statistics – University of São Paulo (USP)
São Paulo – SP – Brazil

{rmcobe,fmresina,renata}@ime.usp.br

Abstract. *Ontologies have been largely used to represent terminological knowledge, specially with the advent of the Semantic Web. As knowledge is not static, it is crucial to learn how to deal with ontology dynamics, which includes Ontology Merging and Debugging. Basically, we want to deal with the inconsistencies and incoherences that may occur when a knowledge base receives a new information or when two or more ontologies are merged. Our purpose in this work is to show some ways to extend and use the BContractor framework, originally proposed for Belief Revision, to implement operations in ontologies.*

1. Introduction

The problem of knowledge representation has been an important object of study for many years, specially in Philosophy, Artificial Intelligence and Cognitive Science. In this context, we find ontologies as an important conceptual model because they provide a formal representation of the domain they describe.

There has been a rapid increase in availability of (semantic) information on the web, i.e., ontologies. Nevertheless, there is no standard way of reusing them, creating a challenge of building new ones. This has forced users to build them from scratch instead of being able to reuse previously established ones. It has been known for years that such reuse would not come for free. The integration of multiple-source ontologies may result in conflicting information being joined together in a single ontology. This kind of problem may compromise the integrity and reliability of an ontology. Such problem is addressed from two main points of view: the theoretical and the pragmatic ones. In the theoretical field, research is more concerned about dealing with logical problems like consistency/coherence checking and solving.

Unfortunately, pragmatic approaches did not follow the same evolution pace and just a few tools have been developed to provide knowledge base integration. Most of them are not able to capture logical problems such as inconsistency and incoherence.

The Belief Revision field deals with the idea that knowledge is not set in stone, i.e., it changes along with time. So, research in the field has been investigating ways to define how an agent should accommodate new information to his/her knowledge base in a consistent way, which leads to the definition of operations to perform this task. Restrictions are given by *r*ationality postulates, which are the properties any operator should obey. Different mathematical constructions are used for the operations. *R*epresentation theorems provide the link between the rationality postulates and the constructions, showing their equivalence.

BContractor [Lundberg et al. 2012] provides a flexible framework for these operations. According to the author's definition, BContractor is "a simple, yet powerful,

interface for operations over belief bases. Simple due to an easily implementable interface. Powerful because it is extensible and easily adapted." The main contribution of this work is to show how we extended the BContractor framework in order for it to work with Description Logics and use it to solve the problem of Belief Revision that emerges from Ontology Merging.

Section 2 brings theoretical background about Belief Revision and Merging. Section 3 introduces the problem of Ontology Merging and the strategies to solve it. Section 4 presents BContractor and the adaptations we developed to extend the framework in order to make it possible to be used with Description Logics and OWL. It includes the application in Merging and the integration with the framework. Section 5 shows a small usage example in Ontology Merging.

2. Belief Revision and Merging

In this section, we briefly present the needed background in Belief Revision and Merging.

Belief Revision deals with the problem of belief dynamics. In this paper, we are interested in the application of this theory in ontology dynamics, i.e., in accommodating new information in a consistent way and also in removing some information from a knowledge base. Most of the studies in this sub-field of knowledge representation are based on the AGM paradigm, whose name derives from the initials of the authors of the seminal paper [Alchourrón et al. 1985]. This paradigm is a theory about how highly idealized rational agents should revise their beliefs when receiving new information.

The epistemic state of an agent can be represented in different ways. In the AGM paradigm the beliefs of an agent are represented by a belief set, i.e., a logically closed set of sentences. So, if K is a belief set, $K = Cn(K)$, where Cn is a supraclassical consequence operator[1]. In addition, from that paradigm we have three main operations regarding a belief set K and a sentence α: expansion (+), contraction (-) and revision(*). We use expansion when we want to simply add a new information to the set ($K + \alpha$). Contraction is used when we want to remove some information ($K - \alpha$) and revision when we want to consistently add a new information to the agent's epistemic state ($K * \alpha$).

In terms of representation, instead of belief sets we are going to represent the epistemic state of the agent by means of belief bases [Hansson 1991], which are sets not necessarily closed under logical consequence. Among the advantages of using this approach, we can cite that working with belief bases is more practical from the computational point of view, considering that belief sets are usually infinite. Moreover, in belief bases we distinguish explicit knowledge from inferred knowledge, exactly because of the absence of logical closure.

In belief base operations, we also work with expansion, contraction and revision. Expansion in bases is defined as $B + \alpha = B \cup \{\alpha\}$. In his paper [Levi 1977], Isaac Levi proposed a process for obtaining the result of a revision by means of a sequence of a contraction and an expansion. Such process became known as the *Levi identity* and works

[1]If $\mathcal{L}$ is a logic closed under the logic connectives ($\wedge, \vee, \neg, \rightarrow$), a consequence operator Cn satisfies supraclassicality if, for any $A \in 2^{\mathcal{L}}$, if α can be derived from A by classical truth-functional logic, then $\alpha \in Cn(A)$

as follows: considering * as a revision function, we formally have $B * \alpha = (B - \neg\alpha) + \alpha$. Therefore, we are going to present here only one operation, contraction. For further details about the relation between contraction and revision, see [Gärdenfors 1988].

In the AGM paradigm, contraction operations are restricted by the so-called rationality postulates. The main constructions found in the literature come equipped with representation theorems. In this paper, we are going to focus on the constructions and their implementation. For details on the postulates and representation results, please refer to [Hansson 1999]. For now, it is enough to know that we want to construct operations that, given a belief base B and a formula α, return a new belief base B' contained in B, that does not imply α and that keep as much information as possible.

In the following, we present two classical constructions for contraction, partial meet contraction [Alchourrón et al. 1985] and kernel contraction [Hansson 1999].

2.1. *Partial Meet* Contraction

A Partial Meet contraction [Alchourrón et al. 1985] of a base B by α consists in finding the maximal subsets of B that do not imply α and take the intersection of a selection of them. For this operation, we need to define the concept of a remainder set $(B \perp \alpha)$:

Definition 1 (Remainder Set) [Alchourrón et al. 1985] *Let B be a belief base and α a sentence. A set B' is an element of the remainder $B \perp \alpha$ if and only if it is a maximal subset of B that does not imply α:*

- *B' is a subset of B ($B' \subseteq B$)*
- *$\alpha \notin Cn(B')$*
- *If $B' \subset B'' \subseteq B$, then $B'' \vdash \alpha$*

An important definition is of a selection function:

Definition 2 (Selection Function) [Alchourrón et al. 1985] *Let $\mathcal{L}$ be a language and B a belief base of this language. For any sentence α, a selection function for B is a function γ such that, for any sentence $\alpha \in \mathcal{L}$:*

- *if $B \perp \alpha \neq \emptyset$, then $\gamma(B \perp \alpha) \neq \emptyset$ and $\gamma(B \perp \alpha) \subseteq B \perp \alpha$Ontology Merging using BContractor: Preliminary Report*
- *if $B \perp \alpha = \emptyset$, $\gamma(B \perp \alpha) = \{K\}$*

Informally speaking, we have the result of the contraction choosing some elements of $B \perp \alpha$ and taking their intersection. Formally:

Definition 3 (Partial Meet Contraction) [Alchourrón et al. 1985] *Let B be a belief base, α an arbitrary sentence and γ a selection function. The Partial Meet contraction function is defined as $B -_\gamma \alpha = \bigcap \gamma(B \perp \alpha)$.*

2.2. *Kernel* Contraction

This construction uses a different approach to solve the problem. Here, the construction consists in finding the minimal subsets of B that imply α and, then, remove at least one element from each of these subsets. The set of these minimal subsets is called the *kernel* of B by α, represented as $B \perp\!\!\!\perp \alpha$.

Definition 4 (Kernel Set) [Hansson 1999] *Let B be a belief base and α a sentence. A set B' is an element of the kernel $B \perp\!\!\!\perp \alpha$ if and only if it is a minimal subset of B that implies α:*

- *B' is a subset of B ($B' \subseteq B$)*
- $\alpha \in Cn(B')$
- *If $B'' \subset B' \subseteq B$, then $B'' \nvdash \alpha$*

An incision function selects at least one element of each kernel to be removed:

Definition 5 (Incision Function) [Hansson 1999] *Let B be a belief base. For any sentence α, an incision function for B is a function σ such that:*

- *$\sigma(B \perp\!\!\!\perp \alpha) \subseteq \bigcup(B \perp\!\!\!\perp \alpha)$ and*
- *if $\emptyset \neq X \in B \perp\!\!\!\perp \alpha$ then $X \cap \sigma(B \perp\!\!\!\perp \alpha) \neq \emptyset$*

A kernel contraction is then defined as removing from the belief base those elements from the kernels selected by the incision function:

Definition 6 [Hansson 1999] *Let B be a belief base, σ an incision function and α a sentence. The Kernel contraction function is defined as $B -_\sigma \alpha = B \backslash \sigma(B \perp\!\!\!\perp \alpha)$*

2.3. Belief Merging and Conflict Resolution

A close related area to Belief Revision is the area of Belief Merging, where instead of adding a single piece of information to a belief base, two (or more) belief bases are combined. In an operation of revision, the incoming information has higher priority over the existing belief base, while in Merging, usually the two belief bases being merged have equal priority. So the areas of Belief Revision and Merging walk hand-in-hand, sharing a large amount of activities to be carried out during the operators executions. For a review on Belief Merging, please refer to [Konieczny and Pérez 2011].

A Merging operator can also be obtained by first joining the two belief bases involved and then solving the conflicts that arise in case the bases are inconsistent. With a slight adaptation, the concepts of remainders and kernels can be used to construct Merging operators. In this paper, we will consider a construction that is based on finding the minimal inconsistent subsets of the joined bases (kernels) and removing at least one element of each.

3. Ontology Merging

When presenting our quick overview of belief revision, we did not specify the logical language used. In fact, the original paper on the AGM paradigm [Alchourrón et al. 1985] does not require a particular logic. Nevertheless, several assumptions are made on the underlying logic, such as supraclassicality, which prevents the paradigm from being directly applicable to several useful logics, such as Description Logics [Baader et al. 2003].

Ontologies describe individuals, classes, attributes of these classes and relationships between them. The OWL language[2], a W3C recommendation since 2004 for representing ontologies, is based on Description Logics - DL. DLs are subsets of First Order Logic, have a well defined semantics and are usually decidable.

[2]http://www.w3.org/TR/owl-guide/

As we can see, if we intend to work with knowledge representation, we should consider these languages for describing ontologies. Nonetheless, in many applications it is not enough to represent knowledge; we should also be able to change it and deal with its dynamics.

There have been several proposals to apply belief revision for ontologies in OWL and DL, such as [Kalyanpur 2006, Ribeiro 2013]. In this paper, we turn to the problem of applying merging operators to combine ontologies and providing an implementation based on *BContractor*.

As mentioned in the Introduction, the integration of multiple knowledge sources will, eventually, result in conflicting knowledge being joined together in a single base. This kind of problem may compromise the integrity and reliability of a knowledge base. When dealing with the integration of ontologies, it is important to distinguish inconsistency from incoherence. An ontological knowledge base is usually divided in two parts: the ABox, containing assertional knowledge about individuals, and the TBox, containing terminological knowledge about concepts and properties. An ontology is considered inconsistent if and only if there is no interpretation that could satisfy all the axioms of the base [Haase et al. 2005]. This kind of problem typically arises with assertional knowledge, i.e., the ABox. A knowledge base is considered incoherent if and only if there is a concept C such that, for all possible models for the knowledge base, C has an empty interpretation [Qi and Pan 2007]. This kind of problem typically arises with terminological knowledge, i.e., the TBox.

Several activities play important roles during the process of inconsistency solving. In [Cobe and Wassermann 2012] the authors group the most common activities developed during the resolution of a conflict into the following phases:

3.1. Kernel Building

The goal of this phase is to build minimal, conflict keeping sub-ontologies, which is closely related to the idea of kernel, i.e., S is a kernel of the inconsistent/incoherent ontology O iff: S is a subset of O, S is inconsistent/incoherent and there is no proper subset of S that is inconsistent/incoherent. We used the same designation as [Kalyanpur 2006, Wassermann 1999]. The concept of kernel is similar to the *Minimal Incoherence Preserving Sub-Ontologies* (MIPS) and *Minimally Unsatisfiability Preserving Sub-TBoxes* (MUPS) [Schlobach 2005, Haase et al. 2005]. In a typical ontology merging scenario, the user might have to examine each kernel at a time, probably using different strategies to deal with each inconsistency/incoherence.

3.2. Stratification

During this phase, the axioms in the chosen kernel are ordered according to some principle - the number of axioms that share concepts and individuals, for instance. We chose to use the same denomination presented in [Qi and Pan 2007, Meyer et al. 2005]. The goal of this phase is similar to the one of the incision functions presented earlier, which also rank axioms according to some criteria. The main difference is that stratification defines strategies to order axioms possibly from one single kernel and incision functions take as input all possible kernels, thus, we can think of incision functions as being composed by stratification strategies - responsible for ordering axioms - and a selection function - which removes the least preferred axiom from every stratified kernel.

The stratification phase can be carried out manually by domain experts [Haase and Volker 2008, Ribeiro and Wassermann 2008], or by automatic means. Now, we are going to enumerate a couple of the most common approaches for stratification.

Specific Axiom Prioritization has been proposed by Qi et al. in [Qi and Pan 2007] and its main idea, taken from [Benferhat 2003], aims to preserve the axioms that describe more general concepts, or more formally: an axiom $\phi_1 = C_1 \sqsubseteq D_1$ is more specific than the axiom $\phi_2 = C_2 \sqsubseteq D_2$ if and only if $C_1 \sqsubseteq C_2$ and $C_2 \not\sqsubseteq C_1$.

Kalyanpur, in [Kalyanpur 2006] also proposed a few algorithms for axiom ranking:

- Order by frequency: which orders by the number of kernels in which the axiom appears;
- Order by semantic relevance: which orders by the number or entailments that are lost or added if the axiom is removed; and
- Order by syntactic relevance: which orders the number of axioms that share the concepts with the axiom being ranked.

All of these strategies are also good candidates for composing incision functions. We implemented some of these approaches using the BContractor framework (see Section 4 below).

3.3. Axiom Weakening

The activities in this phase try to solve the inconsistencies (not incoherences) by modifying the axioms, weakening their restriction power. This phase is not shared with Belief Revision. When we allow the operator to weaken the formula in order to keep consistency we can no longer guarantee that the formula α will be in the resulting knowledge base. This phase is still very useful in cases when the user wants to maintain most of the information in a knowledge base and he/she does not care if the information in the knowledge base is slightly different from before the merging. The goal here is to avoid discarding whole axioms. In what follows, we list some of the main strategies found in the literature.

The first strategy we would like to point is the exception adding, described in [Qi et al. 2006]. The idea consists in transforming inconsistent kernels of the form $K = \{C \sqsubseteq D, C(a) \sqcap \neg D(a)\}$ into $K' = \{(C \sqcap \neg\{a\}) \sqsubseteq D, C(a) \sqcap \neg D(a)\}$.

In [Cobe and Wassermann 2012], the authors proposed a new way of weakening cardinality restrictions. The idea is to iteratively change the value of n in the $\leq nP$ axiom, where n is a number and P a property. They designed a weakening operator that takes all possible minimally inconsistent sets and tries to fix the largest number of inconsistencies by changing the n value. The algorithm proposed needs to check all kernel set because if the inconsistency is fixed in one specific kernel it may be the case that when the ontology is put together, the inconsistency will appear again.

The authors showed that, in this case, an approach closer to believe revision may be better suiting, where an incision function, used to choose which axioms would be weakened - instead of removed -, is composed by a single stratification strategy and is able to select the axioms involved in the conflict which may be more than one axiom from each kernel.

3.4. Axiom Removal

This phase aims to remove the axiom with the lowest priority (or trustability) in the kernel in which the user is working to solve the conflict. This approach is used in most of the works on ontology debugging [Schlobach et al. 2007, Schlobach 2005] and Belief Revision in DL [Ribeiro 2013].

4. BContractor-DL

After some decades of research and study in Belief Revision, we have many results available in the literature, including comparisons between the different possible operators. Nevertheless, there is still a gap when we consider software tools to work with these operators or computational resources analysis. There are some implementations of Belief Revision operators available. As examples, we can cite BReLS [Liberatore 1999] and Saten [Williams and Sims 2000]. However, they focus on a specific logic or construction.

Considering this scenario, the BContractor [Lundberg et al. 2012] was recently released with the purpose of being a more flexible framework for implementing and testing Belief Change operators. One of its main purposes is the possibility of extending it to implement and test operators for different kinds of logic, although so far it has been tested only for Propositional Logic, considering that most work on Belief Revision theory is based on that logic. Still, much effort is being applied to adapt such theory to other logics. Following this purpose, we describe in this section the extensions that we have made to the BContractor in order for it to support DL knowledge bases.

This implementation is restricted to the $\mathcal{SROIQ}$ DL [Horrocks et al. 2006] due to the fact that this DL family is the one that underpins OWL2, which is the language supported by most of the reasoners, including HermiT[3], the one we used. For more expressive DLs, one should use a language more expressive than OWL2 and another reasoner to support it.

The first inclusion was a new component that was needed to calculate kernels from inconsistent knowledge bases. We needed to do that because we are dealing with DL, and as presented in [Ribeiro 2013], in several DLs the negation of an axiom is not defined. So, in order to avoid the usage of negation, instead of revision we rely on the operation of semi-revision presented in [Hansson 1999] and also used in [Ribeiro 2013]. The idea is to insert α in the knowledge base and then contract it by the inconsistency.

The new component defined was a new version of the BContractor *KernelOperator*, the *KernelConflictOperator*, which is able to compute kernels from inconsistent/incoherent knowledge bases. The main difference between the two is that in *KernelConflictOperator* we are able to compute kernels from a knowledge base instead of a knowledge base plus an axiom. These two components only define interfaces and in order to use them, one need to give concrete implementations. We implemented a concrete version of the *KernelConflictOperator* in the *BlackBoxKernelOperator*, so now it is also able to use the *eval* operation on a knowledge base, instead of a pair.

After that, we have built the ground for the definition of Revision operators. We developed the *InternalKernelRevisionWithoutNegation* component, which uses an incision function and a *KernelConflictOperator*. Then it builds a new knowledge base from

[3] `http://hermit-reasoner.com/`

the execution of the incision function in the kernels built by the *KernelConflictOperator*. We use an Internal Revision technique [Ribeiro 2013] in which first we open room for the new piece of knowledge, and then we really add it. The operator behaves as follows: first it calculates the kernels for the knowledge base union α, then it executes the incision function. After that it removes the result of the incision function from the knowledge base and finally it adds α to the knowledge base.

The usage of the BContractor made it really easy to code a new revision operator as shown in Listing 1.

Listing 1. InternalKernelRevisionWithoutNegation Revision Operator

```
1 ISet<S> revise(ISet<S> base, S alpha) {
2   return base.minus(incision(kernel(base.union(alpha)))).union(alpha);
3 }
```

In Listing 1 the *kernel* operation uses the *BlackBoxKernelOperator* to build the kernel set from the union of the knowledge base with *alpha*, then the incision function takes place and calculates a cutting set - a set that contains at least one element of each kernel in the kernel set. After that we can be sure that we removed at least a single element from each kernel breaking their minimality principle, thus restoring their consistency. The result of the incision function is removed from the base with the *minus* operation and only after "making room for *alpha*" is that we include it with the *union* operation.

In addition to that, we have developed two incision functions. The first function prioritizes removing the elements that appear in the largest number of kernels. This function was described by Kalyanpur in [Kalyanpur 2006] and is very useful when we try to keep as much information as possible. So we do not remove a separate axiom from each kernel. In this way we do not remove more axioms than we need to, e.g., consider the following kernel set: $K = \{\{C \sqsubseteq D, C(a) \sqcap \neg D(a)\}, \{C \sqsubseteq D, C(b) \sqcap \neg D(b)\}\}$. By applying this incision function the resulting cutting set would be: $K' = \{C \sqsubseteq D\}$. The implemented component is called *MostFrequentFirstIncisionFunction*.

We have also developed the operator proposed by Qi et al in [Qi and Pan 2007], *MostSpecificFirstIncisionFunction*, that keeps the most general axioms in the knowledge base, e.g., consider the following kernel set: $K = \{\{C \sqsubseteq \neg D, C \sqsubseteq F, F \sqsubseteq D\}\}$. By applying this incision function the resulting cutting set would be $K' = \{C \sqsubseteq \neg D\}$[4].

The usage of these operators is simple, the user has to pass them to the *Revision Operator* being used and call the *revise* operation. In order to use incision functions in a stand-alone way, the user needs to instantiate them and pass the kernel set to their *eval* functions.

As this paper presents an ongoing work the study of the properties of the incision functions developed will be done in the future.

For the Belief Merging case, we developed a new type of operator, the *MergeOperator*, which has a single operation, *merge*, that receives two knowledge bases and produces a new one as output.

[4]Another possible cutting set would be $K'' = \{C \sqsubseteq F\}$. The BContractor-DL chooses only one of the possibilities - the first one.

We have implemented the new idea of *StratificationOperator* which aims to order the kernels according to some specific criteria and also the *WeakenOperator* that builds weaker versions of the kernels in order to restore their consistency. We then developed two stratification operators, the *FrequencyStratificationOperator* and the *GeneralityStratificationOperator* that use the same ideas from the revision incision functions: *MostFrequentFirstIncisionFunction* and *MostSpecificFirstIncisionFunction* respectively.

We have also defined a weakening operator named *NumberedRestrictionWeakenOperator* that uses the idea described in Section 3.3. When the authors presented this approach in [Cobe and Wassermann 2012] they showed that in order to use this wakening strategy we needed a incision function that selects maybe more than one single element from each kernel in the kernel set, e.g., suppose we have the following inconsistent ontology: $O = \{C \sqsubseteq \leq 1P, a_2 \neq a_3, a_2 \neq a_4, a_3 \neq a_4\, C(a_1), P(a_1, a_3), P(a_1, a_2), P(a_1, a_4)\}$. Calculating its kernel set we obtain $K = \{\{C \sqsubseteq \leq 1, C(a_1), a_2 \neq a_3, P(a_1, a_2), P(a_1, a_3)\}, \{C \sqsubseteq \leq 1, C(a_1), a_2 \neq a_4, P(a_1, a_2), P(a_1, a_4)\}, \{C \sqsubseteq \leq 1, C(a_1), a_3 \neq a_4, P(a_1, a_3), P(a_1, a_4)\}\}$. A Numbered Restriction Incision function would return the following cutting set $K' = \{\{C \sqsubseteq \leq 1, P(a_1, a_3), P(a_1, a_2), P(a_1, a_4)\}$. For this matter we developed the *NumberedRestrictionIncisionFunction*.

So, with all these, we have built the ground for the definition of the first merging operator using BContractor, which is defined as the build of a new base from the weakening of the stratified base built from the kernels of the union of the two bases being merged. Using the design ideas from the BContractor, the code for doing so is still human-readable and easy to redefine:

Listing 2. Merge Operator

```
1  ISet<S> merge(ISet<S> base1, ISet<S> base2){
2    Kernel<S> kernelSet = kernel(base1.union(base2));
3    Strata<S> stratifiedKernelSet = stratify(kernelSet);
4    ISet<S> cuttingSetToWeaken = weakenIncision(stratifiedKernelSet);
5    ISet<S> weakenedSet = weaken(cuttingSetToWeaken);
6    if(reasoner.isConsistent(base1.union(base2).minus(
7         cuttingSetToWeaken).union(weakenedSet))){
8           return base1.union(base2).minus(cuttingSetToWeaken).
9             union(weakenedSet);
10   }
11   else{
12     return base1.union(base2).minus(incision(stratifiedKernels));
13   }
14 }
```

The code listed first builds the kernelSet of the union of the two bases (line 2), after that, as explained in [Cobe and Wassermann 2012] we need to use a Most Frequent First [Kalyanpur 2006] strategy to stratify the base. This causes the $\leq nP$ axioms to appear first in the kernel inside the kernel set. This step is important so the *NumberedRestrictionIncisionFunction* knows that all kernels contain the same $\leq nP$ axiom and that that is the axiom to be weakened. After that we use the *NumberedRestrictionIncisionFunction* to select the elements from the kernel set that are going to be weakened. Although only one axiom will be weakened by the *WeakenOperator*, the *NumberedRestrictionIncisionFunction* includes the property assertions that will be used to calculate the new n value, that will be the amount of property assertions.

The *NumberedRestrictionWeakenOperator* is called by the *weaken* function, building a weakened version of the axioms selected by the *NumberedRestrictionIncisionFunction*. After that the *MergeOperator* verifies if the consistency was restored. If that is the case, then it removes the elements selected by the *NumberedRestrictionIncisionFunction* from the base and add their weakened version built by the *NumberedRestrictionWeakenOperator*.

In the worst case scenario, if that does not restore consistency, the *MergeOperator* calculates another cutting set, using a second incision function and removes those axioms from the base, restoring the consistency. The second incision function is needed because it probably will select less elements than the one used for axiom weakening.

5. Usage Example

In this section we are going to describe a small example that aims to show how the user could interact with the framework and how it can be used to restore consistency/coherence in ontologies being merged. The example is very restrict due to lack of space.

Suppose that we have the following ontologies O_1 and O_2 composed by the axioms[5]:

O_1 :
$\phi_1 = E \sqsubseteq F$,
$\phi_2 = F \sqsubseteq \leq 1P$,
$\phi_3 = E(a_1)$,
$\phi_4 = a_2 \neq a_3$,
$\phi_5 = a_2 \neq a_4$,
$\phi_6 = a_3 \neq a_4$,

O_2 :
$\phi_7 = a_2 \neq a_3$,
$\phi_8 = a_2 \neq a_4$,
$\phi_9 = a_3 \neq a_4$,
$\phi_{10} = P(a_1, a_2)$,
$\phi_{11} = P(a_1, a_3)$,
$\phi_{12} = P(a_1, a_4)$

The resulting ontology from the union of O_1 and O_2, $O = \{\phi_1, \phi_2, \phi_3, \phi_7, \phi_8, \phi_9, \phi_{10}, \phi_{11}\ \phi_{12}\}$, is inconsistent, due to the fact that the individual a_1 relates to more than 1 other individual by means of the P property, while the axiom ϕ_2 explicitly says the opposite.

We define the following approach of solving inconsistency after joining the ontologies: we will define a new *MergeOperator* that uses a *BlackBoxKernelOperator*, a *FrequencyStratificationOperator* for stratification, a *NumberedRestrictionIncisionFunction* for selecting axioms to be weakened and a *NumberedRestrictionWeakenOperator* as a weakening operator.

The kernel building operator produces the following kernels, $K_1 = \{\phi_1, \phi_2, \phi_7, \phi_3, \phi_{10}, \phi_{11}\}$, $K_2 = \{\phi_1, \phi_2, \phi_8, \phi_3, \phi_{10}, \phi_{12}\}$ and $K_3 = \{\phi_1, \phi_2, \phi_9, \phi_3, \phi_{11}, \phi_{12}\}$.

When the *FrequencyStratificationOperator* is executed on the kernels obtained previously, it results in the following stratified kernels $K'_1 = \{\phi_1, \phi_2, \phi_3, \phi_{10}, \phi_{11}, \phi_7\}$, $K'_2 = \{\phi_1, \phi_2, \phi_3, \phi_{10}, \phi_{12}, \phi_8\}$ and $K'_3 = \{\phi_1, \phi_2, \phi_3, \phi_{11}, \phi_{12}, \phi_9\}$. The calculated frequency for axioms was 3 for ϕ_1, ϕ_2, ϕ_3, 2 for ϕ_{10}, ϕ_{11}, ϕ_{12} and 1 for ϕ_7, ϕ_8, ϕ_9.

[5]Note that axioms ϕ_4,ϕ_5, ϕ_6 are the same as ϕ_7, ϕ_8, ϕ_9.

The *NumberedRestrictionIncisionFunction* then calculates the cutting set $CS = \{\phi_1, \phi_{10}, \phi_{11}, \text{and } \phi_{12}\}$ that can be used to feed the *NumberedRestrictionWeakenOperator*. The weakening operator execution results in the modification of the axiom ϕ_2 into the axiom $\phi_2' = F \sqsubseteq \leq 3P$ that is used to update the union of the O_1 and O_2 ontologies. Just to give an idea of the syntax, the output of the weakening process is at follows:

```
ClassAssertion(<#E> <#a1>)
SubClassOf(<#F> ObjectMaxCardinality(3 <#P> owl:Thing))
SubClassOf(<#E> <#F>)
DifferentIndividuals(<#a2> <#a3> )
DifferentIndividuals(<#a3> <#a4> )
DifferentIndividuals(<#a2> <#a4> )
ObjectPropertyAssertion(<#P> <#a1> <#a4>)
ObjectPropertyAssertion(<#P> <#a1> <#a3>)
ObjectPropertyAssertion(<#P> <#a1> <#a2>)
```

6. Conclusion

In this paper we showed an extension of the BContractor framework in order to: (a) apply it to Description Logics and (b) implement Merging operators. The extension shows that BContractor is indeed independent of the underlying logics and that it is easily extensible to implement different Belief Change operators, as promised on its release. It permits re-usability of code and more modularized and organized software applications.

The code for the extension of BContractor is freely available at `http://www.ime.usp.br/~rmcobe/OntologyMerging/`. We have also integrated the extension with the Protégé[6] revision plugin first described in [Ribeiro and Wassermann 2008] and available at `https://code.google.com/p/review-and-contract/`.

Future work includes the implementation of the Ontology Merging operators as parts of the Protégé plug-in and empirically testing the different merging strategies on benchmark ontologies. We also plan to study the formal properties of the incision functions implemented and described in Section 4.

Acknowledgments The first and the second authors are supported by the São Paulo Research Foundation (FAPESP), grant numbers 2008/10498-8 and 2011/04477-0, respectively. The third author is partially supported by CNPq, grant number 304043/2010-9. This research is part of FAPESP project OnAIR 2010/19111-9.

References

Alchourrón, C., Gardenfors, P., and Makinson, D. (1985). On the logic of theory change. *Journal of Symbolic Logic*, 50(02):510–530.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook*. Cambridge University Press.

Benferhat, S. (2003). A stratification-based approach for handling conflicts in access control. In *SACMAT'03*, pages 189–195.

Cobe, R. and Wassermann, R. (2012). Ontology merging and conflict resolution. In *Workshop on Belief Change, Non-monotonic Reasoning and Conflict Resolution (BNC)*.

[6]Protégé is a free and open-source ontology editor, serving as a framework for knowledge bases. It was developed by Stanford University, also receiving collaboration from the University of Manchester. Available at `http://protege.stanford.edu/`

Gärdenfors, P. (1988). *Knowledge in Flux - Modeling the Dynamics of Epstemic States*. MIT Press.

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., and Sure, Y. (2005). A framework for handling inconsistency in changing ontologies. In *ISWC' 05*, pages 353–367. Springer.

Haase, P. and Volker, J. (2008). Ontology learning and reasoning — dealing with uncertainty and inconsistency. In *Uncertainty Reasoning for the Semantic Web I*, volume 5327 of *LNCS*, pages 366–384. Springer.

Hansson, S. O. (1991). *Belief Base Dynamics*. PhD thesis, Uppsala University, Suécia.

Hansson, S. O. (1999). *A Textbook of Belief Dynamics*. Kluwer Academic Publishers, Norwell, MA, USA.

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible sroiq. In *KR*, pages 57–67.

Kalyanpur, A. (2006). *Debugging and repair of owl ontologies*. PhD thesis, University of Maryland, College Park, MD, USA.

Konieczny, S. and Pérez, R. P. (2011). Logic based merging. *Journal of Philosophical Logic*, 40(2):239–270.

Levi, I. (1977). Subjunctives, dispositions and chances. *Synthese*, 34(4):423–455.

Liberatore, P. (1999). BReLS: a system for revising, updating, and merging knowledge bases. In *Proceedings of NRAC*.

Lundberg, R., Ribeiro, M., and Wassermann, R. (2012). A framework for empirical evaluation of belief change operators. In *SBIA 2012*, LNAI 7589, pages 12–21. Springer.

Meyer, T., Lee, K., and Booth, R. (2005). Knowledge integration for description logics. In *AAAI'05*, pages 645–650.

Qi, G., Liu, W., and Bell, D. (2006). A revision-based approach to handling inconsistency in description logics. *Artif. Intell. Rev.*, 26:115–128.

Qi, G. and Pan, J. (2007). A stratification-based approach for inconsistency handling in description logics. In *IWOD'07*, page 83, Innsbruck.

Ribeiro, M. and Wassermann, R. (2008). The ontology revisor plug-in for Protégé. In *WONTO*.

Ribeiro, M. M. (2013). *Belief Revision in Non-Classical Logics*, volume XI of *Springerbriefs in Computer Science*. Springer.

Schlobach, S. (2005). Debugging and semantic clarification by pinpointing. In *The Semantic Web: Research and Applications*, LNCS, pages 27–44. Springer.

Schlobach, S., Huang, Z., Cornet, R., and van Harmelen, F. (2007). Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39:317–349.

Wassermann, R. (1999). *Resource-Bounded Belief Revision*. PhD thesis, Universiteit van Amsterdam.

Williams, M.-A. and Sims, A. (2000). Saten: An object-oriented web-based revision and extraction engine. *CoRR*, cs.AI/0003059.