The Ontology Reviser Plug-In for Protégé

Márcio Moretto Ribeiro and Renata Wassermann
{marciomr,renata}@ime.usp.br

Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil

Abstract. This paper presents an ontology reviser plug-in for Protégé. The plug-in implements several belief base contraction and revision operations for expressive Description Logics. The operations can be selected by choosing the desired properties of the outcome from a menu.

1 Introduction

It has been argued that ontologies are necessary for the development of the semantic web [BLHL01]. In the web context, the role of ontologies is to give a formal representation for conceptual knowledge. For this reason in 2004 the W3C [1] has adopted an ontology language (OWL) as web standard. OWL comes in three flavors: OWL-lite, OWL-DL and OWL-full. The latter two of them have their semantics based on the well known Description Logics (DLs) $\mathcal{SHIF}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$ respectively [Hor03].

Because knowledge in the web is not static the dynamic of ontologies must be studied also. The study of ontology dynamic is the main goal of a growing sub-field of the knowledge representation called ontology evolution (see [HS04] for a good overview).

One important approach to deal with knowledge base (KB) dynamics is belief revision. In belief revision three operations are defined: expansion ($B + \alpha$), contraction ($B - \alpha$) and revision ($B * \alpha$). Expansion simply adds a new sentence to the KB, contraction removes a sentence from the consequences of the KB and revision adds a new sentence to the KB maintaining the consistency. Contraction and revision are defined by sets of rationality postulates (postulates that the operation must satisfy). An important result in belief revision is the equivalence between a set of rationality postulates and a specific construction for an operation. This is result is called the representation theorem for the operation and the construction. The most influential work in the belief revision field [AGM85] proved that a set of rationality postulates for contraction/revision is equivalent to three distinct constructions: partial meet contraction/revision, safe contraction/revision and epistemic entrenchment. The framework became known as the AGM paradigm.

[1] http://www.w3.org/2004/OWL/

In the past few years some authors have tried to apply belief revision techniques in description logics. In [Flo06] the author proved that AGM paradigm can not be applied to $\mathcal{SHIF}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$. In [FPA06,RW06] the authors presented two different ways to change the AGM paradigm so that it can be used with $\mathcal{SHIF}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$. Both works deal with belief sets (logically closed sets). Belief sets are usually infinite which makes the problem computationally impractical. The subfield of belief revision called belief base revision deals with the dynamic of belief bases, which are simply sets of sentences (not necessarily closed by logical consequence). [Han97b] presents constructions and postulates for belief base contraction and revision. The constructions for contraction can be applied to the logics we are interested in, as shown in [HW02]. However, revision can not be directly applied because it assumes the logic to be closed under negation. For this reason, in [RW07,RW08] the authors presented constructions and postulates for belief base revision that are compliant to $\mathcal{SHIF}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$.

In this work we are going to present an ontology reviser that implements the operations presented in [RW08]. The program is a plug-in for the ontology editor Protégé 4 [2]. The implementation uses the OWL API [3] and the OWL DL reasoner Pellet [4].

In the following we are going to use Greek lower case letters $\alpha, \beta, \ldots$ for sentences, the upper case letter B possibly with a subscript $B_1, B_2, \ldots$ for sets of sentences (belief bases), upper case letter words $OWL, BIRD, \ldots$ for concepts in a DL and lower case letter words $tweety, hut, \ldots$ for individuals in a DL.

In the next section, we are going to show an example to explain the importance of some postulates for revision. Section 3 will sum up the revision operators that were implemented. In Section 4 we will briefly explain the constructions that are equivalent to these operations. In the last two sections we are going to explain how the constructions were implemented and suggest some future work. Screenshots of the plug-in can be found in the last pages of this paper.

2 Revising an Ontology

Suppose we are developing an ontology and we begin to add axioms to it. First we want to state that there is an owl called "Huh", that owls are birds and that birds fly. As we are writing about birds we think our knowledge base should have knowledge about penguins also. We write that penguins are birds and we end up with the following ontology which will be called B:

[2] http://protege.stanford.edu/
[3] http://owlapi.sourceforge.net/
[4] http://pellet.owldl.com/

$$huh \in OWL$$
$$OWL \sqsubseteq BIRD$$
$$BIRD \sqsubseteq FLY$$
$$PENGUIN \sqsubseteq BIRD$$

Although not explicitly written in our ontology, we can infer that penguins fly. This is because every model that satisfies the axioms $PENGUIN \sqsubseteq BIRD$ and $BIRD \sqsubseteq FLY$ should also satisfy the axiom $PENGUIN \sqsubseteq FLY$. But we know that penguins don't fly and suppose we want to make the revision $B * (PENGUIN \sqsubseteq \neg FLY)$. What should be the result of this revision?

Earlier we said that a revision should add a sentence α, in this case $PENGUIN \sqsubseteq \neg FLY$, to a knowledge base, in this case B. Hence, we must have that $\alpha \in B * \alpha$. This is the first postulate for the revision (*success*).

(Success) $\alpha \in B * \alpha$

The second thing we said about revision is that it should maintain the consistency. The second postulate for revision, which is called *consistency*, states precisely that $B * \alpha$ must be consistent.

(Consistency) $B * \alpha$ is consistent

However, we have not defined what consistency is. In DLs there are two different notions of consistency. The first one states that an ontology B is inconsistent if it has no model. In other words, there is no domain set $\Delta \neq \emptyset$ that satisfies all the axioms. This notion of consistency is closer to classical consistency because it trivializes the base, any sentence can be inferred from an inconsistent B. The second notion of consistency states that, if a knowledge base B is inconsistent, there is some concept in B that is necessarily empty. In this work we will follow [FHP$^+$06] and call this second notion coherence and use the word consistency only for the case that B has no model.

Let us go back to our example. We have that penguins are birds and that birds fly and we want to add that penguins don't fly. If we add this sentence, our ontology would still have a model, but the concept $PENGUIN$ can be inferred to be empty. Hence, the resulting ontology is consistent, but incoherent. One may argue that incoherences are not errors. However, since the concept penguin was explicitly mentioned, the fact that it must be empty normally indicates that there is a modeling error. If we want to deal with this kind of error we must change our second postulate to a *coherence* postulate.

(Coherence) $B * \alpha$ is coherent

Suppose that we meet our well know penguin "Tweety" and we add the sentence $tweety \in PENGUIN$ to our ontology. Now if we add $PENGUIN \sqsubseteq \neg FLY$ to our ontology it will have an inconsistency.

Until now we have stated that $B * \alpha$ must satisfy success and consistency or coherence. However, if these are the only postulates for revision, the revision $B * (PENGUIN \sqsubseteq \neg FLY)$ could be just the single sentence ontology:

$$PENGUIN \sqsubseteq \neg FLY$$

This ontology is consistent and it contains α. So it satisfies success and consistency. Notice, however, that the fact that "Huh" is a owl ($huh \in OWL$) and that owls are birds ($OWL \sqsubseteq BIRD$) are not related to the fact that the original ontology is inconsistent with the sentence added. We would like to maintain this kind of sentences. We need a postulate which states that sentences that are not related, in some sense, with the inconsistency should be kept in the revision. We need a *minimality postulate*. One postulate that guaranties this kind of minimality is *core-retainment*:

(Core-retainment) If $\beta \in B \setminus B * \alpha$ then there is $B' \subseteq B \cup \{\alpha\}$ such that B' is consistent/coherent and $B' \cup \{\beta\}$ is not consistent/coherent.

With this postulate we are forced to keep at least $huh \in OWL$ and $OWL \sqsubseteq BIRD$ from the old ontology. All other sentences are relevant to infer the inconsistency and, hence, may be removed.

Now, assume that we have that penguins are birds, "Tweety" is a penguin, penguins don't fly, birds fly and that in order to fly one must have exactly two wings as the following ontology is showing:

$$\begin{aligned} BIRD &\sqsubseteq FLY \\ PENGUIN &\sqsubseteq BIRD \\ tweety &\in PENGUIN \\ PENGUIN &\sqsubseteq \neg FLY \\ FLY &\sqsubseteq haveWing \leq_2 \sqcap haveWing \geq_2 \end{aligned}$$

Suppose now that we find out that "Tweety" has only one wing and we want to add it to this ontology (α is now $tweety \in haveWing \leq_1 \sqcap haveWing \geq_1$). In this case, the sentences $BIRD \sqsubseteq FLY$ and $FLY \sqsubseteq haveWing \leq_2 \sqcap haveWing \geq_2$ can be removed. What is a little bit strange in this example is that the second sentence is only relevant to infer inconsistency together with the sentence $PENGUIN \sqsubseteq \neg FLY$. The *relevance* postulate tries to capture this intuition.

(Relevance) If $\beta \in B \setminus B * \alpha$ then there is $B * \alpha \subseteq B' \subseteq B \cup \{\alpha\}$ such that B' is consistent/coherent and $B' \cup \{\beta\}$ is not consistent/coherent.

Figure 1 shows an ontology in Protégé representing the situation presented above. Figure 4 shows the ontology after the revision assuming that the user chooses to remove the sentence $BIRD \sqsubseteq FLY$.

3 Belief Base Revision

Belief base revision is the study of the dynamics of belief bases. A belief base is simply a set of sentences. In base revision field three operations for belief bases are defined: contraction, expansion and revision. Expansion is defined as $B + \alpha = B \cup \{\alpha\}$. Contraction and revision are not uniquely defined, but are constrained by sets of postulates. In this work we are going the focus in the postulates for revision. Postulates for revision do not tell us how to construct a revision. Constructions for revision are based on constructions for contraction and they will be informally presented in the next section.

We are going to show the postulates for base revision presented in [RW08]. In [RW08] six operations for revision were presented. Each of them can be defined for coherence and for consequence (twelve operations). Most of the postulates used to define these operations were already presented in the previous section and the rest of them will be presented now.

Until now we have that a revision should add a new sentence (success), in a consistent way (consistency/coherence) and should keep sentences unrelated with the inconsistency (core-retainment/relevance). These postulates do not guaranty that we can not add irrelevant sentences. We need a postulate that guaranties that no irrelevant sentences are added to the base:

(Inclusion) $B * \alpha \subseteq B + \alpha$

Furthermore, notice that, if α is itself inconsistent/incoherent, then success and consistency can not be satisfied at the same time. We need weaker versions of them:

(Weak Success) If α is consistent/coherent then $\alpha \in B * \alpha$

(Weak Consistency) If α is consistent/coherent then $B * \alpha$ is consistent/coherent

For now on success will be called strong success and consistency will be called strong consistency.

We will call kernel revision the revision operations that satisfy core-retainment. Thus, kernel revision with strong success for consistency is characterized by the postulates: strong success, weak consistency, inclusion, core-retainment and a postulate named *pre-expansion*.

(Pre-expansion) $B + \alpha * \alpha = B * \alpha$

The revision operations that satisfy relevance will be called partial meet revision and they also satisfy inclusion and pre-expansion. We can define the following revision operations:

- kernel revision with strong success for consistency
- kernel revision with strong success for coherence
- kernel revision with weak success for consistency
- kernel revision with weak success for coherence

– partial meet revision with strong success for consistency
– partial meet revision with strong success for coherence
– partial meet revision with weak success for consistency
– partial meet revision with weak success for coherence

The other four operations showed in [RW08] are not real revision operations, they are called semi-revisions ($B?\alpha$). A semi-revision [Han97a] was proposed to be used when one does not want to give the input sentence a higher priority. For this reason semi-revisions do not satisfy success. Kernel semi-revision for DLs were studied in [HWKP06]. A kernel semi-revision for consistency is characterized by the postulates: inclusion, core-retainment, strong consistency, pre-expansion and a postulate called *internal exchange*.

(Internal Exchange) If $\alpha, \beta \in B$ then $B?\alpha = B?\beta$

Every semi-revision satisfies internal exchange, inclusion and pre-expansion and we can distinguish the last four operations:

– kernel semi-revision for consistency
– kernel semi-revision for coherence
– partial meet semi-revision for consistency
– partial meet semi-revision for coherence

4 Constructions

In the previous section we presented rationality postulates for revision. In this section we are going to present constructions that are equivalent to those sets of postulates.

We are going to present two kinds of constructions for revision: kernel and partial meet revision. Kernel revisions are equivalent to those sets of postulates that satisfy core-retainment and partial meet revisions are equivalent to those that satisfy relevance. All these constructions first add the new sentence α and then contract the inconsistencies/incoherences. They differ from each other in how they contract the inconsistencies/incoherences.

4.1 Kernel Revisions

Kernel revisions try to contract the inconsistencies/incoherences by finding all minimal inconsistent/incoherent subsets of $B+\alpha$, which is called *kernel* of $B+\alpha$, and then removing at least one element of every one of them.

This construction is equivalent to kernel semi-revision. To satisfy strong success we need to protect the input α. We do that by not choosing α to be removed from B. To satisfy strong consistency/coherence we must protect only the consistent/coherent inputs α.

4.2 Partial Meet Revisions

The strategy adopted by partial meet revision is to find all maximal subsets of $B + \alpha$ that are consistent/coherent, which will be called the *remainder set* of $B+\alpha$. Then we choose some, at least one, of these sets and get their intersection.

Like in the previous section this construction is equivalent to the partial meet semi-revision. To satisfy strong success we also need to protect the input and we do that by choosing every maximal consistent subset of B that contains α. Furthermore, in order to satisfies the strong consistency/coherence we must protect only consistent/coherent inputs.

4.3 Representation Theorems

A representation theorem proves that a set of postulates is equivalent to a construction. This means that, in one hand, the construction satisfies all the postulates in the respective set of postulates. On the other hand, if an operation satisfies every postulate in this set of postulates then it can be constructed by the respective construction.

Every revision presented in section 3 is equivalent to a construction informally presented in 4.1 and 4.2. For example the kernel revision with strong success for consistency of the section 3 is equivalent to the kernel construction that protects the input of the section 4.1.

For more details about these operations, constructions, their representation theorems and proofs see [RW08].

5 Implementation

The difficult part of these constructions is to find the kernel and the remainder set of $B + \alpha$. In this section we are going to show how we implemented these operations. To find the kernel we used the algorithms for ontology debugging presented in [KPSH05,SC03]. To find the remainder set we used Reiter algorithm [Rei87] in the kernel.

[KPSH05] presents two algorithms to find what the authors call justifications for a sentence in an ontology: glass box and black box algorithm.

The glass box algorithm depends on the reasoner. It tries to trace the axioms used to prove the inconsistency. The result is inconsistent/incoherent subset of the ontology. Glass box algorithm is already implemented in the OWL API with Pellet:

```
PelletOptions.USE_TRACING = true;
Reasoner pellet = new Reasoner( OWLManager.createOWLOntologyManager() );
pellet.loadOntology( B );
pellet.getKB().setDoExplanation( true );
```

The resulting subset is not guarantied to be minimal. To guaranty the minimality of this subset we used the black box algorithm. Black box algorithm

(algorithm 1) does not depend on the reasoner. It consists in two parts: the first (expand) adds axioms until the set becomes inconsistent/incoherent. The second part (shrink) removes elements one by one and verifies if the ontology remains inconsistent, if not then the removed axiom must be part of the kernel and so it is left in the ontology. The black box algorithm returns a minimal inconsistent/incoherent subset of the ontology, in other words, an element of the kernel. It can be used alone, but it is much more efficient if it is used with the result of the glass box algorithm forming a hybrid algorithm.

Algorithm 1 Black Box Algorithm

```
BlackBox(B + α)
 1  ▷ First Part (Expand)
 2  B' ← ∅
 3  for β ∈ B + α
 4        do B' ← B' ∪ {β}
 5           if B' is inconsistent/incoherent
 6              then Break
 7  ▷ Second Part (Shrink)
 8  for ϵ ∈ B'
 9        do if B' \ {ϵ} is inconsistent/incoherent
10              then B' ← B' \ {ϵ}
11  return B'
```

To find the other elements of the kernel we implemented the following recursive algorithm:

Algorithm 2 Algorithm to find all the element of the Kernel

```
Kernel(B + α)
1  ▷ This algorithm returns the kernel of B + α
2  if B + α is consistent/coherent
3     then return ∅
4  min ← Hybrid(B + α)
5  B' ← B'{min}
6  for β ∈ min
7        do B' ← B' ∪ kernel(B + α \ {β})
8  return B'
```

Once the kernel is computed, the remainder set can be computed finding the minimal cuts (hitting sets) of the kernel. A cut is a set that contains at

least one element of each set in a class of sets. In [Was00] it was proved that a minimal cut in the kernel corresponds to B minus an element of the remainder set and vice-versa. Hence, the remainder set can be computed using the Reiter's algorithm to find minimal hitting sets of a class of sets [Rei87,GSW89].

5.1 Using the Reviser

When the user opens the revision view inside Protégé, he is given a menu of postulates to choose (see Figure 1). The user has a choice between success, weak success or no success, between core-retainment and relevance as minimality conditions, and between strong consistency, weak consistency, strong coherence and weak coherence. After picking up the desired set of postulates and entering the new sentence at the Input Editor (Figure 1), the user is given the corresponding kernel or remainder set as shown in Figure 3. He can then select at least one sentence from each element of the kernel (Figure 3, left hand side) or select at least one element of the remainder set (Figure 3, right hand side) and press Finish. The ontology is then revised accordingly (Figure 4).

The plug-in also offers the possibility of performing kernel or partial-meet contractions, as can be seen in Figure 2.

6 Conclusion and Future Works

We have presented here a plug-in for Protégé that implements the two operations for contraction and the twelve operations for revision presented in [RW08]. The implementation uses the algorithm presented in [HWKP06] using the OWL API and the pellet reasoner.

At this moment the plug-in is compliant with the build 53 of Protégé 4. We are trying to make it compliant with the last build of Protégé 4. Future works also include testing and improving efficiency of the implementation and testing if it is better to compute remainder sets directly or to use the Reiter's algorithm (as we are doing).

References

[AGM85] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May, 2001.

[FHP+06] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *AAAI*, 2006.

[Flo06] Giorgos Flouris. *On Belief Change and Ontology Evolution*. PhD thesis, University of Crete, 2006.

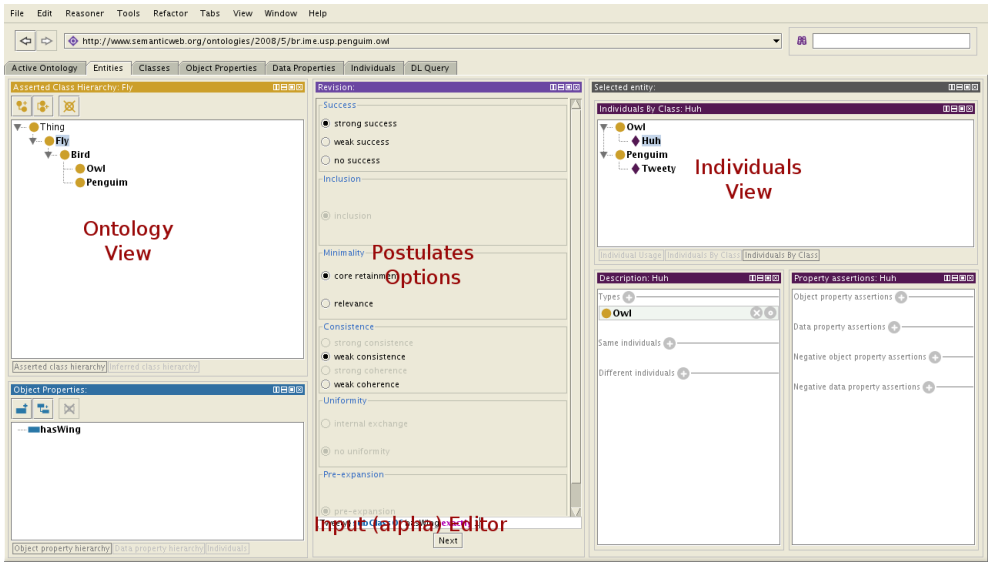

Fig. 1. An overview of a tab of the Protégé with the revision view opened. In the postulate area the user can choose which postulates for revision he wants to be satisfied. In the input editor he should write the axiom to be added.

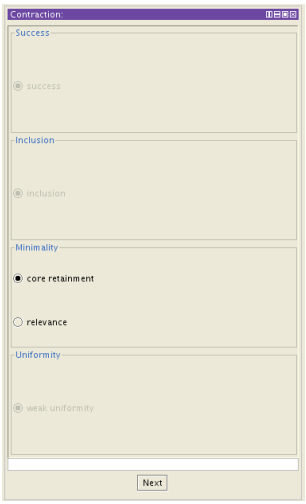

Fig. 2. The contraction view. Like the revision view it has a postulate area and an input editor.

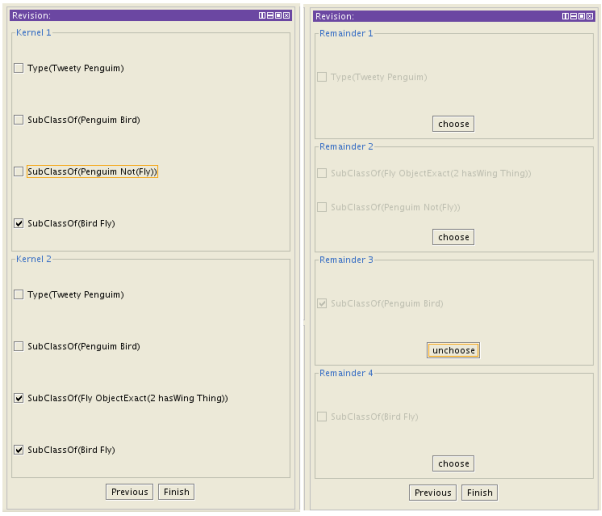

Fig. 3. In the left the elements of the kernel generated by the last example of section 2. In the right elements of the remainder set generated by the same example

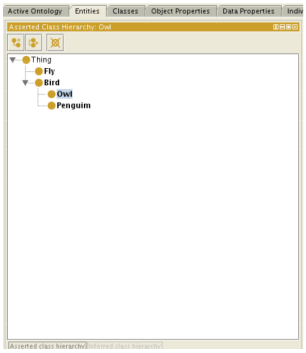

Fig. 4. The ontology updated after the finish button has been pressed in the previous figure.

[FPA06] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On generalizing the agm postulates. In *Proceedings of the 3rd European Starting AI Researcher Symposium (STAIRS-06)*, 2006.

[GSW89] Russel Geriner, Barbara A. Smth, and Ralph W. Wilkerson. A correction to the algorithm in reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.

[Han97a] Sven Ove Hansson. Semi-revision (invited paper). *Journal of Applied Non-Classical Logics*, 7(2), 1997.

[Han97b] Sven Ove Hansson. *A Textbook of Belief Dynamics*. Kluwer Academic Publishers, 1997.

[Hor03] Ian Horrocks. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, 2003.

[HS04] Peter Haase and York Sure. State-of-the-art on ontology evolution. SEKT informal deliverable 3.1.1.b, Institute AIFB, University of Karlsruhe, 2004.

[HW02] Sven Ove Hansson and Renata Wassermann. Local change. *Studia Logica*, 70(1):49–76, 2002.

[HWKP06] Christian Halaschek-Wiener, Yarden Katz, and Bijan Parsia. Belief base revision for expressive description logics. In *OWL: Experiences and Directions 2006*, 2006.

[KPSH05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics - Special Issue of the Semantic Web Track of WWW2005*, 3(4), 2005.

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[RW06] Márcio Moretto Ribeiro and Renata Wassermann. First steps towards revising ontologies. In *Proceedings of the Second Workshop on Ontologies and their Applications (WONTO 2006)*, 2006.

[RW07] Márcio Moretto Ribeiro and Renata Wassermann. Base revision in description logics - preliminary results. In *Proceedings of International Workshop on Ontology Dynamics (IWOD'07)*, JUN 2007.

[RW08] Márcio Moretto Ribeiro and Renata Wassermann. Base revision for ontology debugging. Accepted for publication in the special issue for ontology dynamics in the Journal of Logic and Computation, 2008.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 355–362. Morgan Kaufmann, 2003.

[Was00] Renata Wassermann. An algorithm for belief revision. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann, 2000.