Belief Base Revision For Expressive Description Logics

Christian Halaschek-Wiener[1], Yarden Katz[2], and Bijan Parsia[3]

[1] Department of Computer Science,
University of Maryland, College Park, MD USA,
halasche@cs.umd.edu
[2] Maryland Information and Network Dynamics Lab,
8400 Baltimore Ave., Suite 200, College Park, MD USA,
yarden@umd.edu
[3] School of Computer Science,
The University of Manchester, UK,
bparsia@cs.man.ac.uk

Abstract. In this work, we address the problem of revision of OWL-DL knowledge bases. We focus on belief bases revision as it has previously been shown that OWL-DL is not AGM-compliant for revision. Previously an algorithm for belief base semi-revision for propositional logic has been defined; in particular it has been shown how the diagnosis problem can be translated into a revision problem. In this work, we expand upon this work and detail an approach for performing belief base semi-revision in the Description Logic $\mathcal{SHOIN}$, which corresponds to the W3C standard Web Ontology Language OWL-DL. We additionally, discuss various optimizations to make the approach more practical.

1 Introduction

Recently there has been interest in reasoning with changing Description Logic (DL) knowledge bases. A variety of research directions have been investigated including syntactic updates [8], model-based update semantics (minimal model change) [16, 7], and traditional belief revision approaches based on the AGM postulates [5, 4, 3]. Traditionally, the distinction between update and revision is that in the former the actual state of the world has changed (e.g., via actions of some agent), where as in the latter we incorporate new beliefs about a static world. The choice of which of these types of *change* is appropriate is driven by the application at hand; for purpose of this work, we focus on the problem of belief revision.

The most influential work in belief revision is the AGM model [1], in which the authors define three main change operations on belief states which are represented by logically closed sets of sentences (referred to as *belief sets*). The three operations of change are *expansion* (expanding a belief set with a new belief with no guarantee of consistency after the operation), *contraction* (retracting a belief) and *revision* (expansion with consistency after the operation). Additionally, the authors define a set of postulates for both contraction and revision, which specify what properties a contraction/revision operator must meet in order to be rational.

The basic structure of beliefs assumed by the AGM model is a *belief set*, which is a deductively closed (and hence in general infinite) set of formulae. Due to the difficulty

of computing with belief sets as well as representational issues, there has been substantial work on using *belief bases* as an alternative structure [6, 9, 17, 18]. Belief bases are not closed under logical consequence and are usually interpreted as basic beliefs from which additional beliefs (the belief set) can be derived. In [6, 10, 13] the AGM change operators are defined in terms of belief bases. Additionally, [12] defines *semi-revision*, which differs with the belief base revision model in that the added belief may or may not be accepted (discussed further in Section 2.2).

In this work, we address the problem of semi-revision of finite OWL-DL belief bases. We focus on belief bases as it has previously been shown that OWL-DL is not AGM-compliant for contraction [5, 4, 3] (therefore it can be shown that $\mathcal{SHOIN}$ is not AGM compliant for revision); additionally, one of the biggest short comings in our previous work on syntactic updates [8] is that we do not address resolving inconsistencies after an update. In particular, we extend prior work on belief base revision for propositional logic [21], which allows us to define an algorithm for belief base semi-revisions in the Description Logic $\mathcal{SHOIN}$ (which corresponds to the W3C standard Web Ontology Language OWL-DL). We additionally, discuss various optimizations which make the approach practical.

2 Preliminaries

In this section we provide background information relevant to this paper. First we discuss the Description Logic $\mathcal{SHOIN}$; this is then followed by details regarding belief base revision. Lastly we discuss finding justifications in the Description Logic $\mathcal{SHOIN}$.

2.1 $\mathcal{SHOIN}$ Description Logic

OWL-DL is a syntactic variant of the Description Logic $\mathcal{SHOIN}$ [14], with an OWL-DL ontology corresponding to a $\mathcal{SHOIN}$ knowledge base. In this section, we briefly present the syntax and semantics of the Description Login $\mathcal{SHOIN}$.

Let N_C, N_R, N_I be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles* and *individuals* respectively. The set of $\mathcal{SHOIN}$ roles (roles, for short) is the set $N_R \cup \{R^-|R \in N_R\}$, where R^- denotes the inverse of the atomic role R. Concepts are inductively using the following grammar:

$$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \bowtie nS \mid \{a\}$$

where $A \in N_C$, $a \in N_I$, $C_{(i)}$ a $\mathcal{SHOIN}$ concept, R a role, S a *simple* role [4] and $\bowtie \in \{\leq, \geq\}$. We write $\top$ and $\bot$ to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$ respectively.

A *role inclusion axiom* is an expression of the form $R_1 \sqsubseteq R_2$, where R_1, R_2 are roles. A *transitivity axiom* is an expression of the form $Trans(R)$, where $R \in V_R$. An RBox R is a finite set of role inclusion axioms and transitivity axioms. For C, D concepts, a *concept inclusion axiom* is an expression of the form $C \sqsubseteq D$. A TBox T is a finite set of concept inclusion axioms. An ABox $\mathbf{A}$ is a finite set of concept assertions of the form

[4] See [14] for a precise definition of simple roles

$C(a)$ (where C can be an arbitrary concept expression) and role assertions of the form $R(a, b)$.

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\mathcal{W}, .^{\mathcal{I}})$, where $\mathcal{W}$ is a non-empty set, called the *domain* of the interpretation, and $.^{\mathcal{I}}$ is the *interpretation function*. The interpretation function assigns to $A \in N_C$ a subset of $\mathcal{W}$, to each $R \in N_R$ a subset of of $\mathcal{W} \times \mathcal{W}$ and to each $a \in N_I$ an element of $\mathcal{W}$. The interpretation function is extended to complex roles and concepts as given in [14].

The satisfaction of a $\mathcal{SHOIN}$ axiom α in an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$ is defined as follows: (1) $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$; (2) $\mathcal{I} \models Trans(R)$ iff for every $a, b, c \in \mathcal{W}$, if $(a, b) \in R^{\mathcal{I}}$ and $(b, c) \in R^{\mathcal{I}}$, then $(a, c) \in R^{\mathcal{I}}$; (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; The interpretation $\mathcal{I}$ is a model of the RBox R (respectively of the TBox T) if it satisfies all the axioms in R (respectively T). $\mathcal{I}$ is a model of $\mathsf{K} = (\mathsf{T}, \mathsf{R})$, denoted by $\mathcal{I} \models \mathsf{K}$, iff $\mathcal{I}$ is a model of T and R.

2.2 Belief Base Revision

As discussed earlier, belief bases are sets of formulae which are not closed under logical consequence and are usually interpreted as basic beliefs from which additional beliefs (the belief set) can be derived. [11] introduces a construction for contraction operators for belief bases called *kernel contraction*. The general idea behind this operator is that if at least one element from each α-kernel (which is a minimal subset of a belief base that implies α) is removed from the belief base, then the base will no longer imply α. [11] formally defines the kernel operator as follows:

Definition 1. *[11] The kernel operation $\perp\!\!\!\perp$ is defined such that for any set B of formulas and any formula α, $X \in B \perp\!\!\!\perp \alpha$ iff:*

1. *$X \subseteq B$*
2. *$X \models \alpha$, and*
3. *for all Y, if $Y \subset X$, $Y \not\models \alpha$*

$B \perp\!\!\!\perp \alpha$ is a referred to as a kernel set, and its elements are referred to as α-kernels.

Note that in general, there are arbitrarily many ways to resolve inconsistencies between new information and our current knowledge. For example, if we hold the beliefs that (i) *Every φ is a ψ* and (ii) *w is a φ*, and we revise our knowledge with (iii) *w is not a ψ*, we are faced with an option: we could either retract (i) or retract (ii), therefore a choice must be made. An *incision function*, defined below (originally defined in [11]), determines our choice in such cases; that is it selects the formula to be removed from every α-kernel when we contract α.

Definition 2. *[11] An incision function for B is a function σ such that for any formula α:*

1. *$\sigma(B \perp\!\!\!\perp \alpha) \subseteq \bigcup(B \perp\!\!\!\perp \alpha)$, and*
2. *If $X \neq \emptyset$ and $X \in B \perp\!\!\!\perp \alpha$, then $X \cap \sigma(B \perp\!\!\!\perp \alpha) \neq \emptyset$*

Kernel semi-revision is then defined as follows [12]:

Definition 3. *[12] The kernel semi-revision operator of B based on an incision function σ is denote $?_\sigma$ and defined such that for all sentences α:*

$$B?_\sigma\alpha = B \cup \{\alpha\} \setminus \sigma((B \cup \{\alpha\}) \perp\!\!\!\perp \bot)$$

This can be thought of as a two-step process: we first add α to B, and second, remove inconsistencies in B if there are any. The name "semi-revision" comes from the fact that in our revision process, the formula α that we revise our knowledge with may not be accepted. In other words, α might be removed as part of the second step.

[12] goes on to define the following postulates that must be satisfied for any operator to be considered a kernel semi-revision operator.

Proposition 1. *[12] An operator ? is a kernel semi-revision operator if and only if for all set B of sentences it satisfies the following postulates:*

1. $\bot \notin Cn(B?\alpha)$ *(consistency)*
2. $B?\alpha \subseteq B \cup \{\alpha\}$ *(inclusion)*
3. *If* $\beta \in B \setminus B?\alpha$*, then there is some* $B' \subseteq B \cup \{\alpha\}$ *such that* $\bot \notin Cn(B')$ *and* $\bot \in Cn(B' \cup \{\beta\})$ *(core-retainment)*
4. $(B + \alpha)?\alpha = B?\alpha$ *(pre-expansion)*
5. *If* α, β*, then* $B?\alpha = B?\beta$ *(internal exchange)*

[21] presents an algorithm for belief base semi-revision for propositional logic; in particular, the author shows how the diagnosis problem as described by [19] can be translated into a revision problem.

2.3 Justifications in OWL-DL

Previously, [2] introduced axiom pinpointing for localizing the set of axioms responsible for inconsistencies in DL knowledge bases for the purpose of computing all extension of default theories. Recently, [15] extended this work for the purpose of debugging and repairing OWL ontologies; in [15] the tableau algorithm is slightly modified to maintain the justifications of entailments. Given this work it is possible to find the *justifications* for arbitrary entailments in OWL-DL knowledge bases. [15] formally defines a *justification* as follows:

Definition 4. *[15] Let* $\mathsf{K} \models \alpha$ *where* α *is a sentence. A fragment* $\mathsf{K}' \subseteq \mathsf{K}$ *is a* ***justification*** *for* α *in* K *if* $\mathsf{K}' \models \alpha$*, and* $\mathsf{K}'' \not\models \alpha$ *for every* $\mathsf{K}'' \subset \mathsf{K}'$.

[15] also provides an algorithm for finding all justifications using axiom tracing and Reiter's Hitting Set Tree (HST) algorithm [19] (a similar approach is used in [21] for propositional calculus). The intuition behind the usage of the Hitting Set relies on the fact that, in order to remove an inconsistency from a KB, one needs to remove from KB at least one axiom from each justification for that inconsistency. The approach starts by adding the negation of α and finds an initial justification (subset of K) using axiom tracing [15]. Following this, a hitting set tree is initialized with the initial justification as its root; then it selects an arbitrary axiom (call it *i*) in the root and generates a new

node with an incoming edge whose label corresponds to the removed axiom. The algorithm then tests for consistency with respect to the $\mathsf{K} \setminus \{i\}$. If it is inconsistent, then we obtain another justification for α w.r.t $\mathsf{K} \setminus \{i\}$. The algorithm repeats this process, namely removing an axiom, adding a node, checking consistency and performing axiom tracing until the consistency test turns positive. Due to space limitation, further details are omitted here; however they can be found in [15]. We follow [15] in denoting the set of all justifications for α in a $\mathcal{SHOIN}$ knowledge base K with $\mathsf{JUST}(\alpha, \mathsf{K})$.

3 Belief Base Revision in OWL-DL

As mentioned ealier, [21] presents an algorithm to kernel semi-revision using Reiter's consistency-based diagnosis and minimal hitting sets algorithm [19]. The author uses Reiter's algorithm to compute the minimal set of justifications for the inconsistency in the belief base. Since the state of the art DL reasoners are tableau-based, an analogous approach is needed for computing all minimal justifications for some entailment in an OWL-DL knowledge base. Here we apply the approach presented in [15] to achieve this task. More specifically, we use the approach of [15] to compute the α-kernels (introduced in [11]) of an OWL-DL knowledge base. With this ability, the results of [11] on revising belief bases easily follow for OWL-DL.

3.1 A Kernel Operator for OWL-DL

We now argue that the previously mentioned approach for finding minimal justifications in OWL-DL can be used as a kernel operator. Specifically we show that $\mathsf{JUST}(\alpha, \mathsf{K}) = \bigcup(\mathsf{K} \perp\!\!\!\perp \alpha)$.

Observation 1 *For any $\mathcal{SHOIN}$ knowledge base K and belief α, $\mathsf{JUST}(\alpha, \mathsf{K}) = \bigcup(\mathsf{K} \perp\!\!\!\perp \alpha)$.*

By definition, $\mathsf{JUST}(\alpha, \mathsf{K})$ contains all justifications for $K \models \alpha$. We can easily see that this set satisfies criteria 1, 2, and 3 from Definition 1. Let $j \in \mathsf{JUST}(\alpha, \mathsf{K})$. By definition, $j \subseteq \mathsf{K}$, thereby satisfying criteria 1. Since j justifies α, we have $j \models \alpha$ thereby satisfying 2. Finally, since justifications are minimal, we obtain condition 3, namely $j' \subset j$, $j' \not\models \alpha$. Thus, $JUST(\alpha, \mathsf{K})$ is a kernel operator for $\mathcal{SHOIN}$ knowledge bases.

3.2 A Kernel Semi-Revision Operator for OWL-DL

In this section we formally define a kernel semi-revision operator for OWL-DL knowledge bases. First we must define some incision function; in general such a function will be application specific. For illustration purposes, we follow [21] and define a minimal incision function for OWL-DL.

Definition 5. *In the following we assume all set of formulae are $\mathcal{SHOIN}$ formulae. Given a $\mathcal{SHOIN}$ knowledge base K, a $\mathcal{SHOIN}$–minimal incision function $\sigma_{\mathcal{SHOIN}}$ is a function mapping a set of sets of formulae into a set of formulae, such that for any set of formulae S:*

1. $\sigma_{\mathcal{SHOIN}}(S) \subseteq \bigcup S$
2. *If* $X \neq \emptyset$ *and* $X \in S$ *then* $X \cap \sigma_{\mathcal{SHOIN}}(\mathsf{K}) \neq \emptyset$
3. *If for all* $X \in S$, $X \cap \mathsf{K} \neq \emptyset$, *then* $\sigma_{\mathcal{SHOIN}}(S) \subseteq \mathsf{K}$, *and*
4. $\sigma_{\mathcal{SHOIN}}(S)$ *is the smallest set satisfying conditions 1–3*

Next we define the semi-revision operator for OWL-DL.

Definition 6. *Given a* $\mathcal{SHOIN}$ *knowledge base* K *and formula* α, *we define the operator* $?_{\sigma_{\mathcal{SHOIN}}}$ *such that:*

$$\mathsf{K}?_{\sigma_{\mathcal{SHOIN}}}\alpha = (\mathsf{K} \cup \{\alpha\}) \setminus \sigma_{\mathcal{SHOIN}}(\mathsf{JUST}(\bot, \mathsf{K} \cup \{\alpha\}))$$

It follows from this definition and our previous observation that $?_{\sigma_{\mathcal{SHOIN}}}$ is a kernel semi-revision operator. We obtain the following easy consequence:

Proposition 2. $?_{\sigma_{\mathcal{SHOIN}}}$ *satisfies the kernel semi-revision postulates.*

4 Base Revision Algorithm

The belief base semi-revision algorithm using the previously mentioned techniques is shown below in Figure 1. First the underlying KB is expanded with the update; if the KB is inconsistent after the expansion, then all justifications for the inconsistency are found using the previously discussed approach for computing JUST for OWL-DL KBs. Following this, the applications specific incision function, generally denoted *Incision* here, is used to select the axioms and/or assertions to remove from the updated KB.

```
procedure BaseRevision(K, α):
Input:
   (1) An initial SHOIN knowledge base K
   (2) An update α
 Return:
   Revised knowledge base

K := K ∪ α
if K is inconsisent:
 kernels := JUST(⊥, K)
 K := K \ Incision(kernels)
return K
```

Fig. 1. Pseudo-code for base revision algorithm.

4.1 Performance Issues

In this section we discuss a variety of performance issues with respect to the revision algorithm presented in Figure 1. We discuss each issue and then present possible optimizations.

Incremental Consistency Checking The first step in the algorithm previously presented is rechecking consistency of the KB after it has been expanded with an update. When the underlying KB is very large, this process can be require substantial computation time, even when the updates are trivial; this is related to the inherent complexity of DL reasoning. There has, however, been recent work on incremental consistency checking of the Description Logics $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ under syntactic ABox updates (which correspond directly to belief base expansion) [8].

More specifically, [8] presents an approach for incrementally updating tableau completion graphs under syntactic ABox updates, therefore providing a mechanism to incrementally determine if the updated KB is consistent. DL tableau-based algorithms decide the consistency of an ABox A w.r.t a KB K by trying to construct (an abstraction of) a common model for A and K, called a *completion graph*. Formally, a completion graph for an ABox A with respect to K is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \dot{\neq})$. Each node $x \in \mathcal{V}$ is labeled with a set of concepts $\mathcal{L}(x)$ and each edge $e = \langle x, y \rangle$ with a set $\mathcal{L}(e)$ of role names (the binary predicate $\dot{\neq}$ is used for recording inequalities between nodes). The update algorithm adds new (resp. removes existing for deletions) components (edge and/or nodes) induced by the update to a completion graph; after this, standard tableau completion rules are re-fired to ensure that the model is complete. Therefore the previous completion graph is updated such that if a model exists (i.e, the KB is consistent after the update) a new completion graph will be found. In [8], we observed that the approach demonstrated orders of magnitude performance improvement and performs in real time for a variety of realistic ontologies. This is shown here for varying sized datasets and addition updates from the Lehigh University Benchmark (LUBM)[5] (1, 2 and 3 Universities) in Figure 2. The optimized version of Pellet is denoted by 'Opt.'. It can be observed in Figure 2 that for updates of varying sizes, the algorithm provides dramatic performance improvements and scales for various update sizes. We note here that similar results were observed for deletion updates as well. Further details are omitted here, however they can be found in [8]. It is clear that applying this technique when rechecking consistency after the expansion will provide clear performance improvements. We note that the approach is currently only applicable for ABox updates; however, we are in the progress of extending the approach for TBox updates, as well as for full support of OWL-DL.

Incremental Consistency Checking for Kernel Operators A main performance issue with this approach is related to efficiently computing all α-kernels (i.e., computing all justifications for an entailment). In the naive implementation, computing the minimal hitting sets [15] requires substantial number of consistency checks; this is evident as the algorithm incrementally removes axioms and performs consistency checks at each node in the HST. Figure 3 shows current computation times for finding all justification in a variety of ontologies [15]. It can be seen that in some cases, the algorithm performs quite well on average; in some cases even less than 2 seconds. However in other ontologies, such as Tambis, the approach requires substantially more time. If a revision approach is to be utilized in applications which are time sensitive, then such performance may not be acceptable.

[5] LUBM Homepage; http://swat.cse.lehigh.edu/projects/lubm/

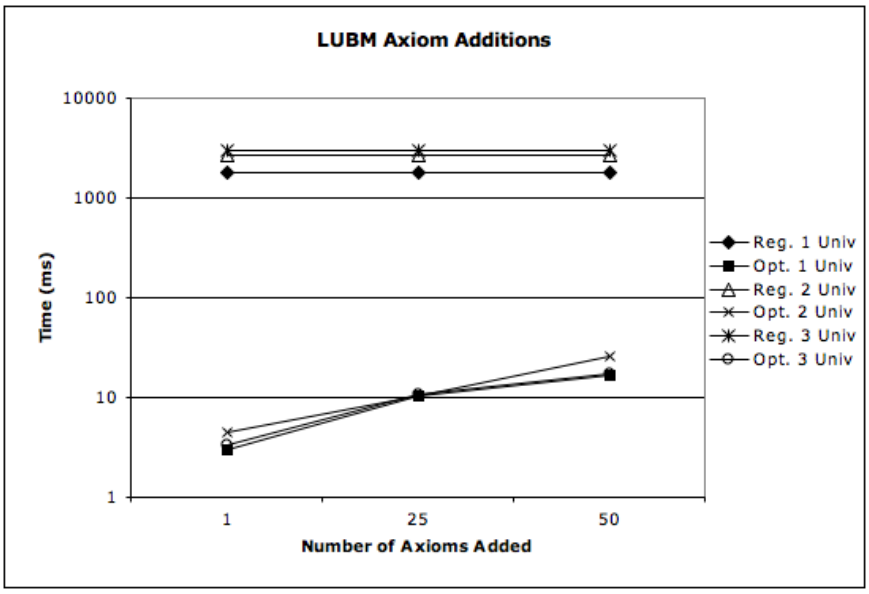

Fig. 2. Addition Updates of LUBM datasets

In order to make this approach more practical it can be noted that our previous work on incremental consistency checking [8] can be leveraged here as well. Rather than checking for consistency from scratch at each iteration of the algorithm to compute all justifications, the incremental approach can be used [15] [6]. More specifically, we can store the completion graph generated by the tableau algorithm at every node of the HST tree and incrementally modifying the graph for every change made (axiom removed). This will clearly save the time required in rechecking consistency from scratch for each new node of the tree.

5 Discussion and Related Work

We briefly point out here that one main limitation of the approach presented in this paper is that the revision approach is syntactic in nature (this is true for all belief base revision operators). That is, the revision performed is dependent on the syntactic form of the explicit axioms in the knowledge base. However, we feel that such an approach is promising and worth investigating as it has previously been shown that the more traditional AGM model is not applicable in a variety of Description Logics, including $\mathcal{SHOIN}$ [5, 4, 3].

There has been a great deal of related work on belief base revision (see [6, 9, 17, 18, 10, 13, 12]); such an approach is more attractive from a computational standpoint than the more general belief set revision [1]. From a Semantic Web standpoint, belief base

[6] Thanks to Aditya Kalyanpur and Vladimir Kolovski for this insight.

Ontology	Base Time (s)	All Justifications (s)	Avg. #Just.	Max. #Just.
Chemical	0.285	1.431	2.8	6
Dolce	0.863	1.034	1	1
Economy	0.179	1.318	1.1	2
Galen	1.232	10.177	1.3	2
Sweet-JPL	0.29	2.541	1.2	2
Tambis	0.434	34.727	3.4	6
Transport	0.59	17.987	2.2	3
University	0.045	0.062	1	1
Wine	0.034	1.137	2.3	5

Fig. 3. Performance for Computing All Justifications in a Variety of Ontologies

revision is preferable for an additional reason, namely a negative result about belief set revision and Description Logics [5, 4, 3]. Recently, there has also been interest in specify formal update semantics for Descriptions Logics [16, 20, 7]; our work here differs in that we address revision of $\mathcal{SHOIN}$ belief bases. Lastly, as previously discussed, [21] presents an algorithm for belief base semi-revision for propositional logic. In this work, we expand upon [21] and provide an approach for performing revisions in the Description Logic $\mathcal{SHOIN}$.

6 Conclusions and Future Work

While well-studied and interesting from a theoretical perspective, belief set revision of the type introduced by the [1] is computationally intractable, making it unrealistic for use in many practical knowledge representation systems. Work on belief base revision attempts to fix these issues by limiting the set of formulae that can be revised. A particularly elegant algorithm for belief base revision, based on Reiter's work on hitting sets for diagnosis, was introduced in [21]. We have shown that an analogue of the technique in [21] for OWL-DL, namely finding of minimal justifications in $\mathcal{SHOIN}$, can be used for revising OWL-DL knowledge bases. Additionally, we have coupled this method with the incremental consistency checking developed in [8], which (based on a preliminary OWL-DL knowledge bases) we hypothesize will out peform a naive belief revision algorithm that recomputes consistency from scratch. For future work, we plan to carry a detailed evaluation of our combined approach for a belief revision system to confirm this hypothesis.

7 Acknowledgments

This work was supported in part by grants from Fujitsu, Lockheed Martin, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial-Intelligence Agency, DARPA, US Army Research Laboratory, and NIST.

We would like to thank Vladimir Kolovski and Aditya Kalyanpur for their contributions to this work.

References

1. Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
2. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
3. G. Flouris, D. Plexousakis, and G. Antoniou. On applying the agm theory to dls and owl. In *4th International Semantic Web Conference (ISWC 2005)*, 2005.
4. G. Flouris, D. Plexousakis, and G. Antoniou. Updating description logic using the agm theory. In *7th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2005.
5. Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Generalizing the agm postulates: Preliminary results and applications. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, Whistler, Canada, June 2004.
6. Andre Furmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20:175–203, 1991.
7. Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On the update of description logic ontologies at the instance level. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, 2006.
8. Christian Halaschek-Wiener, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Description logic reasoning with syntactic updates. In *To appear in Proc. of the 5th Int. Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006)*, 2006.
9. Sven Ove Hansson. New operators for theory change. *Theoria*, 55:114–133, 1989.
10. Sven Ove Hansson. Belief base dynamics. In *PhD Theis, Uppsala University*. 1991.
11. Sven Ove Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59(3):845–859, 1994.
12. Sven Ove Hansson. Semi-revision. *Journal of Applied Non-Classical Logics*, 7(2), 1997.
13. Sven Ove Hansson. A textbook on belief dynamics. Kluwer Academic Press, 1999.
14. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman, 2005.
15. Aditya Kalyanpur. Debugging and repair of owl ontologies. In *Ph.D. Dissertation, University of Maryland, College Park.* http://www.mindswap.org/papers/2006/AdityaThesis-DebuggingOWL.pdf.
16. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *International Conference of Principles of Knowledge Representation and Reasoning(KR)*, 2006.
17. Bernhard Nebel. Syntax-based approaches to belief revision. In P. Gärdenfors, editor, *Belief Revision*, volume 29, pages 52–88. Cambridge University Press, Cambridge, UK, 1992.
18. Bernhard Nebel. How hard is it to revise a belief base? In Didier Dubois and Henri Prade, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 3: Belief Change*, pages 77–145. Kluwer Academic Publishers, Dordrecht, 1998.
19. R Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.
20. Mathieu Roger, Ana Simonet, and Michel Simonet. Toward updates in description logics. In *International Workshop on Knowledge Representation meets Databases*, 2002.
21. R. Wassermann. An algorithm for belief revision. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, 2000.