Trust-Based Revision for Expressive Web Syndication

Jennifer Golbeck[1] and Christian Halaschek-Wiener[2]
[1]College of Information Studies
[2]Department of Computer Science
University of Maryland, College Park
College Park, Maryland 20742
{golbeck,halasche}@cs.umd.edu

Abstract

Interest in web-based syndication systems has been growing as information streams onto the web at an increasing rate. Technologies, like the standard Semantic Web languages RDF and OWL, make it possible to create expressive representations of the content of publications and subscriptions in a syndication framework. Because these languages are based in description logics, this representation allows the application to reasoning to make more precise matching of user interests with published information. A challenge to this approach is that the consistency of the underlying knowledge base must be maintained for these techniques to work. With the frequent addition of information from new publications, it is likely that inconsistencies will arise. There are many potential mechanisms for choosing which inconsistent information to discard from the KB to regain consistency; in the case of news syndication, we argue keeping the most *trusted* information is important for generating the most valuable matches. Thus, in this article, we present algorithms for belief-base revision, and specifically look at the user's trust in the information sources as a metric for deciding what to keep in the KB and what to remove.

1 Introduction

The interest in web-based syndication systems has been growing as information streams onto the web at an increasing rate. In a typical syndication framework, a publisher posts feeds on the web and registers those feeds with a syndication broker. Users, or consumers of the information, register their subscriptions with the broker, who matches the subscriptions to newly published information.

At the same time, technologies for sharing information in a machine processable way have matured on the web. This has increased the power available to publishers to describe their content, to consumers to describe their subscriptions, and to brokers for matching content to subscriptions. Specifically, there has been a transition from keyword based approaches [49] to attribute-value pairs (e.g., [1]) and more recently to XML [3, 13]. Given the limited knowledge modeling expressivity of XML (and XML Schema) there has been interest in using RDF for syndication purposes [57, 9]. RDF has even been adopted as the standard representation format of RSS 1.0[1].

These technologies are still quite limited in their expressive power from a modeling perspective, and in their reasoning capabilities. Semantic Web languages built on top of RDF, particularly the Web Ontology Language (OWL), provide much greater expressivity for describing content and subscriptions. That, in turn, allows finer grained control over queries and content matching [56]. When content and subscriptions are described in OWL, the matching task can be handled using OWL reasoners. While the benefits of a more expressive language

[1]RSS 1.0 Specification: http://web.resource.org/rss/1.0/spec

are clear, there are major challenges to implementing such a system. In particular, reasoning in OWL is difficult. Even when restricted to OWL-DL, the fragment of OWL that aligns with well studied description logics (DL), the time required to match subscriptions and content is prohibitive. Since syndication frameworks make frequent incremental updates to the underlying knowledge base, it is necessary that the reasoner can quickly incorporate these changes. In earlier work [28, 30], we presented a framework for using OWL for representing published content and subscriptions and introduced several optimizations to OWL-DL reasoners to handle the frequent incremental updates. That work is described in more detail in section 2.

There remain unsolved problems that are a barrier to using expressive languages for syndication. As new information is published, it is likely that it will contradict information already in the knowledge base; sources may provide conflicting reports and old information becomes obsolete. Our current techniques can detect these logical inconsistencies, but a method for resolving them is necessary. This requires identifying the source of the inconsistency and deciding what information to keep and what to revise or throw out to regain consistency. The naïve solution would ignore any new statement that makes the knowledge base inconsistent, however this is clearly not a good solution for news updates. We argue that the choice should be made based on trust in a statement, which is a function of the age of the statement and how much trust the broker has in the source of that statement.

Thus, in this paper we present a technique for revising OWL-DL knowledge bases using trust. We begin with a discussion of expressive web syndication and the optimizations that we have developed to provide the necessary reasoning efficiency. We then discuss algorithms for belief-base revision and present an belief-base semi-revision algorithm for OWL-DL. The semi-revision operator defined uses an incision function, which intuitively selects the assertions from the KB that must be retracted in order to regain consistency. Finally, we present an incision function that uses the notion of trust to select which statements to remove such that consistency is maintained.

2 Expressive Syndication on the Web

Increasingly, people and businesses are recognizing the potential benefits of utilizing the vast amounts of frequently updated data available on the web. Web syndication systems offer a straightforward way to discover and aggregate content, and to connect users with the information they need. As interest has grown, there has been a move to more expressive syndication frameworks that offer more precise matches between publications and subscriptions. At the same time, Semantic Web technologies have matured and are better supported. The natural step to provide increasingly expressive syndication is to merge these two ideas, using OWL to describe content and subscriptions, and reasoners to match the two.

To demonstrate the increased expressivity of an OWL-based syndication approach, consider the following example related to the financial domain. Assume that a stock trader is interested in information contained in news articles (or collections of articles) that discuss news about companies that will make their stocks volatile (i.e., they become risky investments). In particular, assume that the trader is interested in any *RiskyCompany* which the trader defines to be a *company* that has a *product* which *causes* an *infection* or *allergic reaction*. Using an XML-based approach, syndication brokers can provide an XML schema which contains an element *RiskyCompany* and such companies can be declared to be this type of element. However, more complex logical definitions and automatic classification of objects cannot be

supported; therefore an XML-based approach cannot accommodate the previous example. If we consider an RDF-based approach, then a syndication broker can model the financial domain using RDF Schema. Using such an approach slightly more complex subscription matches can be obtained, as one can logically infer that a company is a *RiskyCompany* (e.g., based on subclass relationships). However, in an RDF-based approach, complex logical definitions, such as the previously mentioned *RiskyCompany*, are not definable. In contrast, in an OWL-based approach such expressivity is easily provided. For example, the *RiskyCompany* is defined in Table 1.

```
:RiskyCompany a owl:Class;
  owl:intersectionOf (
    [ a owl:Restriction; owl:onProperty :hasProduct;
      owl:someValuesFrom :AdverseEffectProduct ]
    :Company
  ) .
:AdverseEffectProduct a owl:Class;
  owl:intersectionOf (
    [ a owl:Restriction; owl:onProperty :causes;
      owl:someValuesFrom [ owl:unionOf ( :Infection :AllergicReaction ) ] ]
    :Product
  ) .
:causes a owl:ObjectProperty.
:onRecommendation a owl:ObjectProperty.
```

TABLE 1: Illustration of expressivity in OWL-based syndication (given in turtle syntax for clarity).

While the power of this increased expressivity is clear, the inherent difficulty of DL reasoning causes scalability issues [56, 41, 27]. In domains such as financial news feeds, updates come frequently and they must be reasoned over and delivered in near real-time if they are to be useful for tasks like stock trading. However, the static nature of existing DL reasoning techniques requires reasoning on the KB to be performed from scratch for each update: the consistency of the KB must be ensured, queries must be re-evlautated, etc. This negatively impacts the performance results of existing DL-based applications which are analogous to syndication systems, as performance times are in the tens of seconds [56, 41, 27].

To address these issues, we have developed several techniques and optimizations that lead to a significant improvement in performance of DL-reasoning. In this section, we briefly describe those techniques. Full details are available in [29, 28, 30]. First, we discuss the manner in which new publications are integrated into the syndication broker's knowledge base.

2.1 Syntactic Updates

Before introducing the form of updates that are assumed in this work, we briefly introduce some terminology and notation. Specifically, DL KBs are comprised of three main components, namely a *TBox*, *RBox*, and *ABox*. The TBox contains intensional knowledge (axioms about concepts) in the form of a terminology. The axioms in the TBox can be built using the a variety of concept constructors provided in the specific DL, as well as concept inclusion axioms ($\sqsubseteq$), which state inclusion relations between DL concepts. For example, one can state that any technology company is a company via the following axiom: *TechnologyCompany* $\sqsubseteq$ *Company*. The RBox contains intentional knowledge about the roles in the knowledge base. For example, one can state that any two individuals that satisfy the *hasCEO* role,

also satisfy the *hasEmployee* role by the following axiom: *hasCEO* $\sqsubseteq$ *hasEmployee*, where *hasCEO* and *hasEmployee* are roles. In contrast, the ABox contains extensional knowledge that is specific to the individuals in the domain of discourse. Assertions in the ABox take the form of concept assertions (e.g., *Company(BauschAndLomb)*), role assertions (e.g., *hasProduct(BauschAndLomb, Renu)*), and equality (*Ford* = *FordMotorCompnay*) and inequality (e.g., *CitiGroup* $\neq$ *CapitalCityBankGroup*) assertions.

In our earlier work, we assume that a publication is represented as a set of ABox assertions [28, 30]. Therefore, when a new publication is received, the ABox assertions must be integrated into the broker's KB. Further, we adopt a specific mechanism for performing this task, which we refer to as *syntactic updates*. Intuitively, this is simply an update in which all new assertions are directly added (or removed) to the asserted (base) axioms; thus, the only changes that occur are those explicitly stated in the update. Formally, this update is described as follows:

Definition 2.1
Let A be the ABox of KB K. Then under *syntactic updates*, updating K with an ABox addition (resp. deletion) β, written as $\mathsf{K}+\beta$ (resp. $\mathsf{K}-\beta$), results in an updated set of ABox axioms A' such that $A' = A \cup \beta$ (resp. $A' = A \setminus \beta$).

It is possible that after syntactic updates the KB may be inconsistent. The revision techniques we introduce in this paper will choose which statements to discard and which to keep in order to regain consistency.

That this type of ABox update is different when compared to related work in update semantics [43] and belief revision [16] for DLs. Syntactic updates has been adopted as these works have unfortunately found negative results when trying to apply leading theories for handling updates in DL KBs, as they are either not representable in OWL DL [43] or they cannot satisfy the requirements imposed by the theory [16]. Further, this type of update is clearly applicable to syndication applications.

2.2 Incremental Consistency Checking

When a new publication is received, the consistency of the KB must be checked 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 $\mathcal{SHOQ}$ (which correspond to large portions of OWL-DL) under syntactic ABox updates [29, 30].

More specifically, [29] 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 with respect to a KB K by trying to construct an abstraction of a common model for A and K, called a *completion graph*. The update algorithm adds new (resp. removes existing for deletions) components (nodes, edges, and/or labels) 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 complete and clash-free completion graph will be found. In [29, 30], we observed that the approach demonstrated orders of magnitude performance improvement and performs in real time for a variety of realistic ontologies.

2.3 Continuous Query Answering

After guaranteeing the consistency of the knowledge base after updates, the subscriptions registered with the broker must be evaluated. Traditionally, this would require reasoning from scratch over all instances in the KB. However, two observations provide insight into how this reasoning can be optimized. First, by monotonicity, when an addition is made to the KB, the query answering reduces to determining any new bindings that are entailed. When there are deletions from the KB, the query answering reduces to guaranteeing that all previous bindings are still entailed. Here, we describe our approach for efficient continuous query answering. Full details are given in [28, 30].

The first step is to localize the effects of an update. A key insight is demonstrated if we consider a simple query such as $\langle x \rangle \leftarrow \text{Company}(x)$. Intuitively, in the event an update is an addition, we would only like to consider affected named individuals not previously bound to x as a potential answers; in contrast if the update is a deletion, only individuals previously bound to x that were affected by the update need to be re-checked. The individuals affected are those which, in *some* model, the update causes them to instantiate a concept or role filler (the reverse holds for deletions). The task of localizing the affected individuals under an update can then be reduced to updating tableau completion graphs.

When queries contain complex patterns, such as role restrictions, it is not enough to consider only directly affected individuals. An unaffected individual connected by a role to an affected individual may also be a candidate answer. Therefore, we also consider the structure of the query when choosing individuals. The key insight is that for a query to have an answer after an update, at least one individual bound to the query must be affected by the update.

In [28, 30], we show that this approach leads to a speedup of several orders of magnitude.

3 Belief Base Revision

The techniques described in section 2 provide the speedups necessary to implement a web-syndication framework using OWL. However, they do not address how to handle inconsistencies that arise in the KB from updates. In this section, we describe belief base revision and how it is generally applicable for resolving inconsistencies after updates.

Belief revision is the process of changing beliefs after new information is received. The most influential work in belief revision is the AGM model [2], in which the authors define three main change operations on belief states which are represented by logically closed sets of statements (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. Expansion is analogous to the manner in which addition updates are handled under syntactic updates.

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 [19, 31, 45, 46]. 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 [19, 32, 35] the AGM change operators are defined

in terms of belief bases. Additionally, [34] 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 3.1).

We address the application of semi-revision to finite OWL-DL belief bases. We choose to focus on belief bases because it has previously been shown that OWL-DL is not AGM-compliant for contraction [18, 17, 16]. We extend prior work on belief base revision for propositional logic [58], which allows us to define an algorithm for belief base semi-revision in the DL $\mathcal{SHOIN}$ (which corresponds to OWL-DL).

3.1 Belief Base Revision Fundamentals

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. [33] 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. [33] formally defines the kernel operator as follows:

Definition 3.1
[33] The *kernel operation* $\perp\!\!\!\perp$ is defined such that for any set B of formulae 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 [33]), determines our choice in such cases; that is it selects the formula to be removed from every α-kernel when we contract α.

Definition 3.2
[33] 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 [34]:

Definition 3.3
[34] The *kernel semi-revision operator* for B based on an incision function σ is denoted $?_\sigma$ and defined such that for all statements 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 the 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.

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

PROPOSITION 3.4
[34] An operator ? is a kernel semi-revision operator if and only if for all set B of statements it satisfies the following postulates:

1. $\perp \notin Cn(B?\alpha)$ (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 $\perp \notin Cn(B')$ and $\perp \in Cn(B' \cup \{\beta\})$ (core-retainment)
4. $(B + \alpha)?\alpha = B?\alpha$ (pre-expansion)
5. If α, β, then $B?\alpha = B?\beta$ (internal exchange)

3.2 Justifications in OWL-DL

Previously, [4] 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, [36] extended this work for the purpose of debugging and repairing OWL ontologies; in [36] 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. [36] formally defines a *justification* as follows:

DEFINITION 3.5
[36] Let $\mathsf{K} \models \alpha$ where α is a statement. 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 $\mathsf{K}'' \subset \mathsf{K}'$.

[36] also provides an algorithm for finding all justifications using axiom tracing and Reiter's Hitting Set Tree (HST) algorithm [53] (a similar approach is used in [58] for propositional calculus). The intuition behind the usage of the HST 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 [36]. 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 [36]. We follow [36] in denoting the set of all justifications for α in a $\mathcal{SHOIN}$ knowledge base K with $\mathsf{JUST}(\alpha, \mathsf{K})$.

3.3 Belief Base Revision in OWL-DL

As mentioned ealier, [58] presents an algorithm to kernel semi-revision using Reiter's consistency-based diagnosis and minimal hitting sets algorithm [53]. 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 [36] to achieve this task. More specifically, we use the approach of [36] to compute the α-kernels (introduced in [33]) of an OWL-DL knowledge base. With this ability, the results of [33] on revising belief bases easily follow for OWL-DL.

Here, we show that $\mathsf{JUST}(\alpha, \mathsf{K}) = \bigcup(\mathsf{K} \perp\!\!\!\perp \alpha)$.

Proposition 3.6
For any $\mathcal{SHOIN}$ knowledge base K and belief α, $\mathsf{JUST}(\alpha, \mathsf{K}) = \bigcup(\mathsf{K} \perp\!\!\!\perp \alpha)$.

Proof. By definition, $\mathsf{JUST}(\alpha, \mathsf{K})$ contains all justifications for $K \models \alpha$. It suffices to show that criteria 1, 2, and 3 from Definition 3.1 are satisfied. Let $j \in \mathsf{JUST}(\alpha, \mathsf{K})$. By definition, $j \subseteq \mathsf{K}$, thereby satisfying criteria 1. Further, by definition $j \models \alpha$; thus criteria 2 is satisfied. Finally, since justifications are minimal, we obtain condition 3, namely $j' \subset j$, $j' \not\models \alpha$. Thus, $\mathsf{JUST}(\alpha, \mathsf{K})$ is a kernel operator for $\mathcal{SHOIN}$ KBs. ∎

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 3.1. 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.

Next we define the semi-revision operator.

Definition 3.7
Given a knowledge base K and formula α, we define the operator $?_\sigma$ such that:

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

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

Proposition 3.8
$?_\sigma$ satisfies the kernel semi-revision postulates.

4 Resolving Inconsistencies Using Trust

Having shown that belief base revision is an effective and appropriate strategy for resolving inconsistencies in OWL-DL knowledge bases, the final step is to define an appropriate incision function. For the kinds of applications that we envision relying on expressive web syndication, such as traders processing financial news, the technical details of the reasoner are hidden and irrelevant; the users simply want the best information available. When a contradiction arises, it is important that the inconsistency be resolved in a way that keeps the best, most *trusted* information in the knowledge base.

If we know the source for each statement (i.e. the publisher), then we can use trust in the source to reflect how much we trust the statements they provide. The link between trust in a source and the statements made by that source is intuitive. We have shown in previous work

that using trust in a source to reflect trust in the information it provides is effective for sorting information [20], aggregating numeric data [21], and prioritizing rules in default logics [37]. Here we extend this line of work to rank statements in our knowledge base.

In this section, we introduce the concept of trust as it applies to information on the web. In particular, we propose an incision function that chooses statements based on the trustworthiness of their sources and how long the statements have been in the KB. Based on this, we present a semi-revision operator and an analysis of the efficiency of this approach.

4.1 Defining and Using Trust

In this application, there are two types of trust. First, we have trust in information providers. These would be the media outlets and individuals who are publishing their content to the web. Secondly, we have trust in individual statements in the KB. Certainly, the trust we have in the source of a statement will influence the trust we have in the statement itself. Time also impacts trust in statements; an old fact, even from the most trusted source, may still become obsolete. Thus, over time the trust in a statement will degrade, eventually allowing more up to date statements into the KB. We begin by outlining the sociological foundations of trust to show its applicability to this problem, and then introduce a function for determining trust in statements.

4.1.1 Trust in Sources

[12] contains a frequently referenced definition of trust. He states that trusting behavior occurs when a person (say Alice) encounters a situation where she perceives an ambiguous path. The result of following the path can be good or bad, and the occurrence of the good or bad result is contingent on the action of another person (say Bob). Furthermore, the negative impact of the bad result is greater than the positive impact of the good result. This further motivates Alice to make the correct choice. If Alice chooses to go down the path, she has made a trusting choice. She trusts that Bob will take the steps necessary to ensure the good outcome. The requirement that the bad outcome must have greater negative implications than the good outcome has positive implications has been countered in other work [23], which does not always require disparity.

[55] presents and justifies a simple, general definition of trust similar to that of Deutsch: "Trust is a bet about the future contingent actions of others." There are two main components of this definition: belief and commitment. First, a person believes that the trusted person will act in a certain way. The belief alone, however, is not enough to say there is trust. Trust occurs when that belief is used as the foundation for making a commitment to a particular action. These two components are also present in the core of Deutsch's definition: we commit to take the ambiguous path if we believe that the trusted person will take the action that will produce the good outcome.

Taking the main social aspects of the definitions above, we propose the following definition for our use: trust in a source is a commitment to an action based on a belief that the source's future actions will lead to a good outcome. Our commitment is to include information from a trusted source over that from an untrusted source in the knowledge base, based on the belief that the source will provide information that will improve the accuracy of our knowledge base. Intuitively, this social definition of trust parallels the way people think about trusting information sources.

Of course, this requires that we know how much to trust the sources. In our syndication framework, the broker is responsible for reasoning and must maintain a trust value for each source. In the news domain where major media outlets are involved, it is likely that the number of sources is small enough that a broker can manually decide how much each should be trusted. In other applications, for example, when posts from blogs are being considered, there are potentially thousands of sources. In these situations, a broker can rely on algorithms that compute trust automatically. In our previous work we have developed algorithms for computing trust through social networks [20, 38], or through similarity measures [61, 22]. Either of these approaches could be used with relatively little bootstrapping by the broker and produce quite accurate results.

4.1.2 Trust in Statements

Assuming we know the source of each statement, the trust we have in that source is a good measure of how much trust we have in the statements asserted by that source. For example, a person who trusts traditional news sources will trust news if it comes from Dow Jones more than if it comes from Bob's Financial News Blog. Note, however, that a broker can make its own trust ratings, and could decide that non-traditional sources may be more trustworthy than major media outlets.

However, for news syndication, a straightforward ranking by trust does not fully capture the complexity of the situation. In particular, since the state of the world is always changing, statements from a highly trusted source will still become obsolete over time. Time is an important component of trust in frequently updated information systems. Thus, trust in a statement must be a function of both the recency of the statement and the trust in its source.

For a statement α, let τ_{source_α} be the user defined trust in the source of α and ρ_α be a measure of the recency of a statement. Trust in α, τ_α, is determined by a combination of the trust in its source and it recency. Higher trust in the source or a more recent statement will be more trusted. This is formally defined in Definition 4.1; note that the notation $source_\alpha$ and $source\prime_\alpha$ denote different sources of α, whereas ρ_α and $\rho\prime_\alpha$ denotes publications of α with different measures of recency (i.e., the greater ρ_α, the longer the information has existed).

Definition 4.1
Trust in α, τ_α, is given by a function ζ such that:

1. $\tau_\alpha = \zeta(\tau_{source_\alpha}, \rho_\alpha)$
2. $\zeta(\tau_{source_\alpha}, \rho_\alpha) < \zeta(\tau_{source\prime_\alpha}, \rho_\alpha)$ if $\tau_{source_\alpha} < \tau_{source\prime_\alpha}$
3. $\zeta(\tau_{source_\alpha}, \rho_\alpha) < \zeta(\tau_{source_\alpha}, \rho\prime_\alpha)$ if $\rho\prime_\alpha < \rho_\alpha$
4. if $\zeta(\tau_{source_\alpha}, \rho_\alpha) < \zeta(\tau_{source_\beta}, \rho_\beta)$ then $\zeta(\tau_{source_\alpha}, \rho_\alpha + i) < \zeta(\tau_{source_\beta}, \rho_\beta + i)$

Condition 4 requires that ρ degrades τ linearly. Consider the case where either α or β must be removed to regain consistency in the knowledge base. Without condition 4, it could be the case that $\tau_\alpha < \tau_\beta$ at time t_1, and thus α would be removed from the knowledge base, but at time t_2, $\tau_\alpha > \tau_\beta$ and we would prefer to have kept α over β. Condition 4 ensures that our choice at time t_1 remains the same at all times t_{1+i}.

The exact balance between recency and trust in the source will vary from one domain to another. We discuss methods for optimizing ζ in section 8. In any optimization, ζ is a straightforward combination of variables, running in O(1) time, and so we will simply use the τ_α variable whenever ζ must be computed.

4.2 A Kernel Semi-Revision Operator using Trust

In this section, we formally define a kernel semi-revision operator with a trust-based incision function. First, the notion of a trust-based selection function is defined.

Definition 4.2
Define a *trust-based selection function* λ as any function mapping a set of $\mathcal{SHOIN}$ assertions into an assertion α, such that for any set X of $\mathcal{SHOIN}$ assertions:

1. If $X \neq \emptyset$ then $\alpha \in X$
2. For each $\gamma \in X$, $\tau_\alpha \leq \tau_\gamma$

Intuitively, the selection function returns (one of) the least trusted assertion. Given this, a trust-based incision function is defined in Definition 4.3.

Definition 4.3
Given KB K, a *trust incision function* σ_τ is any function mapping a set of sets of assertions into a set of assertions, such that for any set S of sets of assertions:

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

It is a direct consequence of the definition that the trust-based incision function satisfies the condition of a general incision function defined in Definition 3.2. In Algorithm 1, we give the algorithm for σ_τ.

Algorithm 1 *Incision*(S)

Input:
S: Set of kernels
Output:
R: Set of formulae to discard

```
1:  R ← ∅
2:  for all X ∈ S do
3:      discard ← null
4:      for all x ∈ X do
5:          if τ_x < τ_discard then
6:              discard = x
7:          end if
8:      end for
9:      R ← R ∪ {discard}
10: end for
11: return R
```

In the worst case the incision function σ_τ must look at each statement in each kernel once, and perform several constant-time operations on it. Therefore, the incision function does not increase the complexity of the belief revision process.

Clearly there can be situations in which the intersection of two kernels A, B is non-empty. In this case, it could be that assertion $\alpha \in A \cap B$ and α is the least trustworthy assertion in A, however not in B. Clearly α will be removed from A, as it is necessary to regain consistency; similarly the least-trusted assertion in B will be retracted as well. This demonstrates that the trust-based incision function is not necessarily a minimal-incision function, as it could be possible to retract a smaller set of assertions and still regain consistency. However, this could easily be taken into account by a slight modification of Algorithm 1, by skipping any kernel X when a statement in X is already in R.

Next we define the semi-revision operator using trust.

Definition 4.4
Given a knowledge base K and formula α, we define the operator $?_{\sigma_\tau}$ such that:

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

It follows from this definition and our previous observation that $?_{\sigma_\tau}$ is a kernel semi-revision operator; therefore, we obtain the following consequence:

Proposition 4.5
$?_{\sigma_\tau}$ satisfies the kernel semi-revision postulates.

5 Base Revision Algorithm

The belief base semi-revision algorithm using the previously mentioned techniques is shown below in Table 2. 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 trust incision function σ_τ is used to select the axioms and/or assertions to remove from the updated KB.

Algorithm 2 *BaseRevision*(K, α)

Input:
K: Initial $\mathcal{SHOIN}$ knowledge base
β: An addition update
Output:
K: Revised knowledge base

```
1: K ← K + β
2: if K ⊨ ⊥ then
3:     kernels ← JUST(⊥, K)
4:     K ← K \ σ_τ(kernels)
5: end if
6: return K
```

In section 2, we showed several techniques that improve the speed of DL reasoning to support the frequent updates of a syndication system. As we add this procedure for resolving inconsistencies, it is important that it performs as well as the other techniques so that it does not become a bottleneck.

A main question of performance 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 [36] 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. However, our previous work on incremental consistency checking [29] improves the performance for finding α-kernels in the same way it improves the performance of consistency checks. Specifically, we 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 saves the time required in rechecking consistency from scratch for each new node of the tree.

The other addition made in this approach is the provenance for each statement. Provenance, or the history of data, can be simple annotations or complex interconnections. In our case, we simply need a record of the source and timestamp for each statement. While no OWL reasoners have built in provenance tracking at this point, it is straightforward to store this information in an external data structure. For our purposes, a Java hash is sufficient and provides efficient (constant time) access.

6 Example

In this section, a brief example of the trust-based revision approach is provided. First, assume that trust values range over the positive integers from 1 to 10. Additionally, for simplicity assume that given an assertion α, the trust function is defined as follows:

$$\zeta(\tau_{source_\alpha}, \rho_\alpha) = \frac{\tau_{source_\alpha}}{\rho_\alpha + 1}$$

That is, the trust in α is simply the result of dividing the trust value for a given assertion by one plus the number of time units since it was published; note that the time since publication is incremented by one to ensure division by a non-zero number.

For ease of exposition, simply assume that the broker's initial KB is empty. Additionally, let there be two publishers, P_1 and P_2, registered with a syndication broker, such that P_1 is a highly trusted source with trust value of 9. Further, P_2 is a far less trust-worthy publisher and is therefore trusted with a value of 3.

Now, say that at time 1, P_1 publishes the assertion *RiskyCompany(BauschAndLomb)*. Given that this is the only assertion in the KB at time 1, the KB is clearly consistent. Next assume that at time 2, publisher P_2 publishes the assertion $\neg$*RiskyCompany(BauschAndLomb)*, clearly resulting in an inconsistency. Using the previously described revision approach, the justifications (constituting the kernels) for $\mathsf{K} \models \bot$ will be determined. In this simple example, there is only one justification composed of the two assertions. Next, the following trust values would be found for the assertions at time 2:

$$\tau_{RiskyCompany(BauschAndLomb)} = \frac{9}{2}$$
$$\tau_{\neg RiskyCompany(BauschAndLomb)} = \frac{3}{1}$$

Given this, the assertion $\neg$*RiskyCompany(BauschAndLomb)* would be removed and consistency would be regained.

7 Related Work

7.1 Syndication Systems

This overview demonstrates the recent trend toward more expressive syndication approaches.

7.1.1 Keywords and Attributes-Value Pairs

Early syndication systems primarily relied on subject-based keywords in order to match user interests with published documents/data [49, 59]. In such an approach, publishers associate some number of predefined keywords with publications, and similarly, subscription are represented using keywords. Therefore, matching publications with subscription requests therefore reduces to keyword matching. The main limitation of such an approach is its simplistic use of keyword matching.

Following this, there were efforts to capture more detailed user interests by allowing attribute-value pairs to be associated with published content [1, 14, 10, 6, 7, 14]. In such an approach users are allowed to provide values for certain attributes, and matching reduces to determining the satisfaction of attribute values with associated publications. While this provides further filtering capabilities and more accurate dissemination, the approach is still limited by the predefined attributes and the simple matching of attribute values.

In order to address the scalability of both of these two approaches, distributed architectures have been investigated (e.g., [5, 59]). For example, [59] presents the following approach: assume there are n distributed brokers and that published documents can be sent to a subset of these distributed brokers, denoted P_B. Similarly, a subscription is assumed to be registered with some subset of the distributed brokers, denoted S_B. Therefore, if $P_B \cap S_B \neq \emptyset$ then it is guaranteed that matches will not be missed (this will always hold if $P_B + S_B = n + 1$) [59]. Using such an approach, users will obtain results as soon as the first broker processes it and the load is distributed.

7.1.2 XML Approaches

Over the past few years there has been vast interest in utilizing XML for filtering purposes in syndication systems (e.g., see [3, 8, 25, 39, 13]). In such an approach, published documents/data are represented in XML and subscription requests are specified using an XML query/path language (e.g., XPath [8]). This approach provides a variety of benefits including the enables the ability to enforce published content validation (using XML Schema) and provides more expressive subscription requests (via XML path query languages), etc. The popularity of such approaches is exemplified by the number of RSS 2.0 [42] and Atom 1.0 [48, 24] feeds (both of which are XML based), which are growing at ever increasing rates.

Recently, there has been work in addressing the scalability of such approaches by proposing distributed XML-based syndication architectures (e.g., see [13, 60]). For example, in [13] the authors investigate a variety of distribution techniques, including distributing queries based on query pattern exclusiveness. The general idea is to require different brokers to be responsible for different queries such that the probability that newly published content satisfies queries at any two brokers is low. For example, two queries that contain value-based predicates on the same attribute but with different values are distributed to different brokers. [13] additionally leverages content-driven routing to direct publications to distributed brokers which are potentially *interested* in the content (i.e., contain subscriptions which will likely match the publication). This is accomplished by maintaining query routing tables which summarize the interests (i.e., subscriptions maintained) of other brokers.

7.1.3 Semantic Approaches

Given the limited expressivity of XML (and XML Schema) as a knowledge modeling language, there has been interest in using formal knowledge representation languages for representing published contents. One such approach is provided in the event-based dissemination platform CREAM [10]; in this platform, the syndication model is attribute-value pair based, however attributes can be associated with semantic information described in an ontology. This allows the definition of concept hierarchies and therefore simple inferencing to determine subscription matches.

Given the recent standardization of the knowledge representation language RDF (and RDFS), there has been increased interest in using RDF [40] for syndication purposes; RDF is even the underlying representation format of RSS 1.0. A variety of architectures have been proposed (e.g., [51, 52, 57]), all of which use RDF the representation language for publications. In such approaches, RDF graph-based query languages (typically triple patterns) are used to represent subscription requests and matching publications with subscriptions reduces to triple pattern matching. Additionally, in most approaches RDFS is also utilized to describe domain ontologies which the published RDF content adheres to. This allows simple semantic inferences (e.g., via subclass relationships) to be made over publications, similar to the approach adopted in CREAM.

There has also been work on addressing the scalability of an RDF-based approaches by leveraging distributed syndication archictectures. For example, [9] present an distributed RDF publish/subscribe approach, in which a peer-to-peer (P2P) architecture is utilized. Specifically, a super-peer [11, 47] architecture is proposed, where super-peers in the network are responsible transfering publications to the correct subscribers, which are regular peers in the network.

Concept-Based Approaches. In [56, 41] the authors use a DL-based approach for Web service matching and information syndication respectively, in which DL concepts (possibly complex concepts) are used to represent both subscription requests as well as published documents/data. In such an approach, matching published content with subscription requests reduces to determining if published concepts and subscriptions are logically equivalent, subsume one another, or are not compatible (discussed below). More specifically, [41, 50] defines there to be a match between a subscription S and published document D if one of the following holds (in order of match strength):

- *Exact* ($S \equiv D$): The subscription and the published concepts are equivalent concepts, and it is referred to as an *Exact* match.
- *PlugIn* ($S \sqsubset D$): The subscription is a sub-concept of the published concept, and it is referred to as a *PlugIn* match.
- *Subsume* ($D \sqsubseteq S$): The subscription is a super-concept of the published concept, and it is referred to as a *Subsume* match.
- *Intersection* ($\neg(S \sqcap D \sqsubseteq \bot)$): The intersection of the subscription and the published concept is satisfiable, and it is referred to as an *Intersection* match.

Matching subscription requests is therefore accomplished by two reasoning tasks; namely classifying the KB when either subscription requests or new data is received and by performing concept satisfiability tests for the intersection of each subscription and published concept. While empirical results demonstrate accepted performance times (~20 ms) for matching new subscription requests with a fixed set of published documents, processing a large amount of

incoming published content is still problematic (~10s of seconds) [56, 41]. This is due to the cost of repeatedly reclassifying the published concepts; this problem is compounded if syndication brokers have domain knowledge (potentially very large) with which newly published data must be integrated.

Query-Based Approach. [26, 27] presents an agent-based document retrieval system (which is essentially a publish/subscribe application) in which published contents are represented as ABox assertions and subscription requests as a DL concept (viewed as an instance retrieval query). Therefore, matching is reduced to instance retrieval of the subscription concept. Given the performance issues of using such an approach (i.e., response times in the 10s of seconds), the authors introduce two optimizations for more effective incremental instance retrieval, both or which are discussed below:

- *Query Ordering*: In query ordering, a partial ordering is induced upon all registered subscription concepts based on their subsumption relations; these subsumption relations are determined by classifying the atomic subscription concepts. In the approach more general subscriptions are answered first, thereby reducing the number of individuals that must be considered for more specific queries (due to the subsumption relation).
- *Candidate Individual Reduction*: Candidate individual reduction is the process of not considering previous individuals which satisfied registered queries; that is, once an individual has satisfied the subscription concept, it does not need to be reconsiders when new data is published. This holds, due to the monotonicity of the DLs considered in the work and the fact that deletions are not supported.

7.1.4 Discussion

Our work is based in using a more expressive mechanism for representing published contents. This allows the use of automated reasoning procedures to infer matches not found using syntactic approaches (keyword, attribute-value pair, and/or XML) and simpler semantic approaches (e.g., RDF/S). While there has been work on application scenarios which are related to DL-based syndication systems, the related techniques either take a different approach for representing published contents and subscriptions requests (e.g., [56, 41]) or assume a simpler subscription format with only atomic concepts (e.g., [26, 27]). Further, in the syndication framework used in this work, we assume the publications are encoded in OWL, which provides additional benefits.

Another general distinction between the syndication framework used here and those presented above is that our approach allows for publications to persist at the syndication broker for varying time frames; this in turn allows composite subscription matches, in which the information contained in multiple publications comprises a publication match.

7.2 Belief Revision

There have been numerous approaches investigated in literature for revising and updating logical knowledge bases. Recently, there have been attempts to apply these approaches to description logic knowledge bases.

As discussed in section 3, one of the main problems with the AGM model of belief revision is related to the difficulty of of computing with belief sets. Given this, there has been substantial work on using *belief bases* as an alternative structure [44, 19, 31, 45, 46]. 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 [19, 32, 35] the traditional AGM change operators are defined in terms of belief bases.

There has been recent work on applying traditional belief base revision algorithms to description logics. In [15], the author follows the base revision approach presented in [19] and again finds negative results, in that $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ cannot satisfy the belief base revision postulates originally presented in [19].

Given these negative results, there has been recent interest in applying belief base *semi-revision* [34] to DL KBs. Semi-revision differs with the traditional belief base revision model in that the added belief may or may not be accepted (this is because it may be later retracted in order to regain consistency). In literature, semi-revision has successfully been applied to propositional logic. For example, [58] presents an algorithm for belief base semi-revision for propositional logic based on the construction presented in [34], and the author shows how the diagnosis problem as described by [53] can be used to provide an semi-revision algorithm for propositional logic.

There has been recent work on applying semi-revision techniques to description logics. [54] presents two different constructions based on semi-revision, both of which aim to ensure the that the new belief is entailed by the revised KB. In the first approach, it is guaranteed that the new belief will be entailed by the knowledge base, unless that new belief is inconsistent. In the second construction, the new belief will always be entailed after revision, however if the new belief is inconsistent then the revised knowledge base is inconsistent as well. The main distinction with the approach presented in this article is that, while it is ensured that the KB is consistent after the revision, in our approach the new belief may not be entailed by the revised KB.

8 Conclusions and Future Work

As interest in web syndication grows, and more applications begin to rely on syndicated data, there is an increased need for more expressive ways to describe content and subscriptions. We have created a syndication framework using the Semantic Web language OWL, and introduced a series of reasoning optimizations that allow a system to handle the rapid flow of information. This framework needed an efficient and effective method for resolving inconsistencies in the knowledge base. In this paper, we have presented a trust-based approach to belief base revision, and shown it applicability to OWL-DL knowledge bases.

There are several directions of future work in this project. We are in the process of evaluating our syndication system using real-time news feeds in the financial domain. These simulations will allow us to demonstrate the effectiveness and scalability of the entire system, including the belief base revision techniques presented here.

There are also several trust questions we will address independently. The exact balance between τ_{source_α} and ρ_α is likely application specific. We plan to run experiments with our existing syndication data to develop a method for optimizing each factor in ζ. Additionally, there are some questions to be investigated regarding the trust incision function σ_τ, outlined in table 1. That algorithm will choose a set of statements to discard such that the KB will be consistent. However, that set is not necessarily minimal or optimal. For example, it could be the case that one statement, s_a is in every α-kernel, and is the least trusted in some, but not all, α-kernels. Depending on the ordering, σ_τ could end up selecting only s_1 for removal, or s_1 in addition to several other statements. It is unclear what an "optimal" set would be,

but there are certainly optimizations that can be made to the incision function. We plan to investigate this topic, with attention to trust, information in the knowledge base, as well as the complexity of the incision function.

9 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 Aditya Kalyanpur, Yarden Katz, Vladimir Kolovski, and Bijan Parsia for their contributions to this work.

References

[1] Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *Symposium on Principles of Distributed Computing*, 1999.

[2] 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.

[3] Mehmet Altinel and Michael J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *The VLDB Journal*, 2000.

[4] F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.

[5] Guruduth Banavar, Tushar Deepak Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Stur. An efficient multicast protocol for content-based publish-subscribe systems. In *ICDCS '99: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, page 262, Washington, DC, USA, 1999. IEEE Computer Society.

[6] A. Carzaniga, M. Rutherford, and A. Wolf. A routing scheme for content-based networking. In *Technical Report CU-CS-953-03, Department of Computer Science, University of Colorado, June 2003.*, 2003.

[7] A. Carzaniga and A. Wolf. Forwarding in a content-based network. In *In SIGCOMM '03, Karlsruhe, Germany, Aug.*, 2003.

[8] Chee Yong Chan, Pascal Felber, Minos N. Garofalakis, and Rajeev Rastogi. Efficient filtering of XML documents with XPath expressions. *The VLDB Journal*, 11:354–379, 2002.

[9] Paul Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl. Publish/subscribe for rdf-based p2p networks. In *In Proceedings of 1st European Semantic Web Symposium.*, 2004.

[10] M. Cilia, C. Bornhvd, and A. P. Buchmann. Cream: An infrastructure for distributed, heterogeneous event-based applications. In *Proceedings of the International Conference on Cooperative Information Systems*, 2003.

[11] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.

[12] Morton Deutsch. *The Resolution of Conflict*. Yale University Press, New Haven, 1973.

[13] Y. Diao, S. Rizvi, and M. Franklin. Towards an internet-scale xml dissemination service. In *Proceedings of VLDB2004, August 2004.*, 2004.

[14] Françoise Fabret, H. Arno Jacobsen, François Llirbat, João Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):115–126, 2001.

[15] G. Flouris. On belief change and ontology evolution. In *Ph.D. Dissertation, University of Crete*, 2006.

[16] 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.

[17] 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.

[18] 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.

[19] Andre Furmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20:175–203, 1991.

[20] Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, College Park, MD, USA, April 2005.

[21] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the Fourth International Conference on Trust Management*, 2006.

[22] Jennifer Golbeck. Trust and nuanced profile similarity in online social networks. In *MINDSWAP Technical Report TR-MS1284*, 2006.

[23] Robert T. Golembiewski and Mark McConkie. The centrality of interpersonal trust in group processes. In Cary Cooper, editor, *Theories of Group Processes*. Wiley, Hoboken, NJ, 1975.

[24] J. Gregorio and B. de hra. The atom publishing protocol. In *IETF Internet Draft*, 2005.

[25] Ashish Kumar Gupta and Dan Suciu. Stream processing of xpath queries with predicates. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 419–430, New York, NY, USA, 2003. ACM Press.

[26] V. Haarslev and R. Moller. Description logic systems with concrete domains: Applications for the semantic web. In *In Int. Workshop on KR meets Databases, 2003.*, 2003.

[27] V. Haarslev and R. Möller. Incremental query answering for implementing document retrieval services. In *Proc. of the Int. Workshop on Description Logics*, 2003.

[28] Christian Halaschek-Weiner and James Hendler. Toward expressive syndication on the web. In *Proceedings of the 16th World Wide Web Conference*, 2007.

[29] Christian Halaschek-Wiener, Bijan Parsia, and Evren Sirin. Description logic reasoning with syntactic updates. In *Proc. of the Int. Conf. on Ontologies, Databases, and Applications of Semantics*, 2006.

[30] F. Christian Halaschek-Wiener. *Expressive Syndication on the Web Using a Description Logic Based Approach*. PhD thesis, University of Maryland, College Park, 2007. http://www.mindswap.org/~chris/publications/FCHW-Thesis.pdf.

[31] Sven Ove Hansson. New operators for theory change. *Theoria*, 55:114–133, 1989.

[32] Sven Ove Hansson. Belief base dynamics. In *PhD Theis, Uppsala University*. Uppsala University, 1991.

[33] Sven Ove Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59(3):845–859, 1994.

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

[35] Sven Ove Hansson. *A Textbook on Belief Dynamics*. Kluwer Academic Press, 1999.

[36] Aditya Kalyanpur. Debugging and repair of owl ontologies. In *Ph.D. Dissertation, University of Maryland, College Park*, 2006. http://www.mindswap.org/papers/2006/AdityaThesis-DebuggingOWL.pdf.

[37] Yarden Katz and Jennifer Golbeck. Social network-based trust in prioritized default logic. *Proceedings of the Twenty-First National Conference onArtificial Intelligence (AAAI-06)*, 2006.

[38] Ugur Kuter and Jennifer Golbeck. Sunny: A new algorithm for trust inference in social networks, using probabilistic confidence models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[39] Laks V. S. Lakshmanan and Sailaja Parthasarathy. On efficient matching of streaming XML documents and queries. In *Extending Database Technology*, pages 142–160, 2002.

[40] O. Lassila and R.R. Swick. Resource description framework (rdf) model and syntax specification. w3c recommendation, www consortium (cambridge, ma). February 1999.

[41] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International World Wide Web Conference*, 2003.

[42] D. Libby. Rss 0.91 spec, revision 3. In *Netscape Comm.*, 1999.

[43] 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.

[44] Bernhard Nebel. A knowledge level analysis of belief revision. In R. Brachman, H. J. Levesque, and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, pages 301–311, San Mateo, 1989. Morgan Kaufmann.

[45] 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.

[46] 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.

[47] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, and Alexander Loser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference*, 2003.

[48] M. Nottingham and R. Sayre. The atom syndication format. In *IETF Internet Draft*, 2005.

[49] B. Oki, M. Pfluegl, and Dale Skeen. The information bus: An architecture for extensible distributed systems. In *In Proc. 14th SOSP*, 1993.

[50] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *The First International Semantic Web Conference*, 2002.

[51] Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. S-topss: Semantic toronto publish/subscribe system. In *VLDB '03: Proceedings of the 29th international conference on Very large data bases*, 2003.

[52] Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen. Cms-topss: Efficient dissemination of rss documents. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1279–1282. VLDB Endowment, 2005.

[53] R Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[54] Marcio Moretto Ribeiro and Renata Wasserman. Base revision in description logics - preliminary results. In *Proc. of the Int. Workshop on Ontology Dynamics (IWOD 2007)*, 2007.

[55] Piotr Sztompka. *Trust: A Sociological Theory*. Cambridge University Press, Cambridge, 1999.

[56] Michael Uschold, Peter Clark, Fred Dickey, Casey Fung, Sonia Smith, Stephen Uczekaj Michael Wilke, Sean Bechhofer, and Ian Horrocks. A semantic infosphere. In *Proc. of the Int. Semantic Web Conference*, 2003.

[57] Jinling Wang, Beihong Jin, and Jing Li. An ontology-based publish/subscribe system. In *Middleware*, 2004.

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

[59] Tak W. Yan and Hector Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.

[60] Eiko Yoneki and Jean Bacon. Distributed multicast grouping for publish/subscribe over mobile ad hoc networks. In *Wireless Communications and Networking Conference*, pages 2293–2299, 2005.

[61] Cai-Nicolas Ziegler and Jennifer Golbeck. Investigating Correlations of Trust and Interest Similarity. *Decision Support Services*, page 1, 2006.