The Meaning of Erasing in RDF under the Katsuno-Mendelzon Approach

In Memory of Alberto O. Mendelzon

Claudio Gutierrez[c] Carlos Hurtado[c] Alejandro Vaisman[c]
[c]Department of Computer Science, Universidad de Chile
{cgutierr,churtado,avaisman}@dcc.uchile.cl

ABSTRACT

The basic data model for the Semantic Web is RDF. In this paper we address updates in RDF. It is known that the semantics of updates for data models becomes unclear when the model turns, even slightly, more general than a simple relational structure. Using the framework of Katsuno-Mendelzon, we define a semantics for updates in RDF. Particularly we explore the behavior of this semantics for the "erase" operator (which in general is not expressible in RDF). Our results include a proposal of sound semantics for RDF updates, a characterization of the maximal RDF graph which captures exactly all consequences of the erase operation expressible in RDF, and complexity results about the computation of this graph and updates in RDF in general.

1. INTRODUCTION

The *Semantic Web* is a proposal oriented to represent Web content in an easily machine-processable way. The basic layer of the data representation for the Semantic Web recommended by the World Wide Web Consortium (W3C) is the Resource Description Framework (RDF) [12]. The RDF model is more than a simple relational structure; its expressivity turns more general the existential conjunctive fragment of first order logic by adding transitivity of some predicates and inheritance axioms. From a database point of view, it can be viewed as an extension of a representation system along the lines of naive tables without negation [1].

In this paper we concentrate on the problem of updating RDF data. In the last two years the semantic web community has shown an increasing interest in this problem. However, the existing proposals have so far ignored the semantic problems associated to the presence of blank nodes and of RDFS vocabulary with built-in semantics [15, 17, 22, 14], and tackled the subject from a syntactical point of view. Related to the update problem in RDF, some studies have addressed changes in an ontology [13, 19], and more recently, the representation and querying of temporal information in RDF [6] has been also studied.

Supported by Millennium Nucleus Center for Web Research, Grant P04-67-F, Mideplan, Chile, and the Project *Procesamiento y Analisis Semantico de Servicios Web*, Fondecyt No. 1050642.

1.1 The Problem of Updates in RDF

Updates and Revision. The semantics of updates for data models becomes difficult when the model turns –even slightly– more general than a simple relational structure [4]. For knowledge bases, the abstract general problem of updating is: *what should be the result of changing a theory T with a sentence φ?* As Katsuno and Mendelzon [10] argued, the answer to this problem depends on the application at hand. There is a fundamental distinction between *update* (now in a technical sense) and *revision* [11, 10]. *Update* means bringing the knowledge base up to date when the world described by it changes; *revise* means incorporating new information obtained about a static world. This discussion is relevant when facing updates in the RDF model. Thus, the distinction between update and revision becomes of central importance. On the one hand, one of the main design goals of the RDF model is allowing distributed revisions of the knowledge base in the form of addition of information in a monotonic way [20]. By some classic results of Gardenförs [3], the notion of revision becomes trivial in this setting. On the other hand, when viewing RDF from a database point of view (i.e., huge but delimited repositories of metadata, like metadata for a library, for instance), the notion of update becomes relevant. In this paper we concentrate on this latter notion, and follow the approach of Katsuno and Mendelzon [10].

Updates in RDF. Management, in particular maintainability, of RDF data needs a well defined notion of update. The problem becomes relevant since the standardization of a query language for RDF [16]. We will show that the problem of characterizing these changes in RDF is far from being trivial and raises interesting practical and theoretical issues that, so far, have been overlooked.

Consider for example the case of a web music store that uses Semantic Web technology for making it easier to find information about artists depending on their music styles. This is a very dynamic environment, where artists and music styles are continuously being updated. Figure 1 shows a small portion of this web site, where `sc` means "subclassOf", `type` indicates an instance of a class, and an edge between two nodes represents a triple of the form, for instance $(a, \texttt{sc}, c)$. Suppose we want to delete all triples containing the value *artist* in Figure 1 (a). The result, clearly, is the one shown in Figure 1 (b), where dashed lines indicate

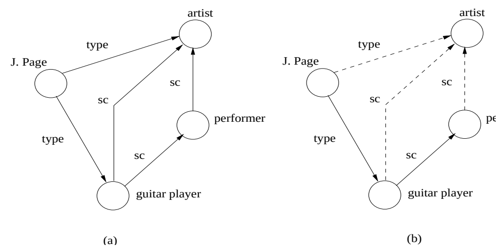

Figure 1: Deleting all triples containing "artist".

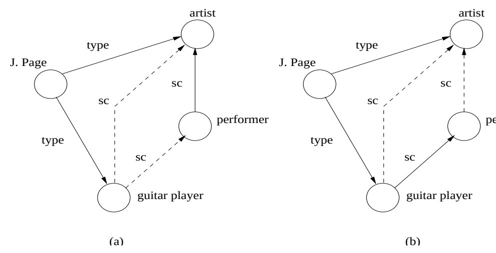

Figure 2: Deleting the triple ($guitarplayer$, $\texttt{sc}$, $artist$).

the deleted arcs and nodes. However, if we want to delete the triple ($guitarplayer$, $\texttt{sc}$, $artist$), a reasonable semantics for this operation must ensure that the triple above cannot be deduced from the updated database. This semantics yields two possible results, depicted in Figures 2 (a) and (b). Additionally, we have to decide what to do with the triple ($J.Page$, $\texttt{type}$, $artist$): was it inserted directly, or was deduced from the triples ($J.Page$, $\texttt{type}$, $guitar\ player$) and ($guitar\ player$, $\texttt{sc}$, $artist$)? (see Section 3). In the former case, it should stay; in the latter it should be deleted. What is to be done? Expressing the new scenario is beyond the expressivity of RDF. One of the goals of this paper is to give a sound semantics for this operation. In this version, we concentrate on ground graphs (i.e., RDF graphs without blank nodes) and the operation of erase. In this direction, we characterize the formulas expressible in RDF which remain logical consequences of a graph G after erasing from it another graph H.

The paper is organized as follows. In Section 2 we discuss related work. Section 3 reviews RDF concepts and presents a formalization of RDF. In Section 4 we introduce our semantics for updates based on the Katsuno-Mendelzon approach. Section 5 presents a characterization of erasing in RDF. Section 6 studies the complexity of the update and erase operations proposed. We conclude in Section 7.

2. RELATED WORK

Updates in knowledge bases and representation systems. The semantics of an incomplete database (i.e. a relational databases containing incomplete information) is the set of all of its possible states. Updates are then defined over this interpretation. Thus, a deletion would consist in eliminating a tuple from every possible database state. Analogously, an insertion must be applied to all possible states. The notion of *representation system* comes in to determine the degree in which the system is capable of expressing the new state of the database. In short, a representation system is composed of a set of tables, a mapping from tables to instances, and a set of allowed operations (like insertion, join, and so on). If the exact result of all allowed expressions can be computed, we have a strong representation system. Otherwise, we may limit to obtain approximate answers (and we have a weak representation system). A result by Imielinsky and Lipski [9] states that representation systems based on naive tables (a relation containing variables and constants) are *weak* for the standard relational operations not including negative selection nor set difference. In [1] this result is extended to consider updates. They show that, for naive tables, adding the insertion operation yields a *weak* representation system. However, if Ω contains positive selection, projection, and deletion, we do *not* have a weak representation system. This result is explained by the fact that naive tables do not handle disjunction. The conclusion is that naive tables are adequate for querying but not for updates. As RDF can be considered an extension of a representation system based on the notion of naive tables without negation, we conclude that, in order to be appropriate for handling *update* and *erase*, RDF would need negation and disjunction.

Updates in graph databases. Updates have been also studied in the context of graph databases. This is relevant to our work because the RDF model is closely related to graph data models [2]. In particular, the Graph-based data model (GDM) and its update language GUL, introduced by Hidders [8] are based on pattern matching. Two basic operations are defined in GUL: *addition* and *deletion.* In the case of *deletion*, there is a base pattern which contains a *core* pattern. The nodes, edges and class names that are not in core pattern are deleted for every matching of the base pattern. This approach is a promising line to implement in RDF the semantic notions presented in this paper.

Updates in web databases: XML and RDF. XML Updates have been extensively addressed in the XML world. Tatarinov *et al* [18] proposed an XQuery extension that has been the first step leading to the proposal currently under study at the W3C [21]. The W3C specified the properties required for update operators in XML. RDF Updates have recently attracted the attention of the RDF community. Nevertheless, all proposals have so far ignored the semantic problems arising for updates associated to the existence of blank nodes and the presence of RDFS vocabulary with built-in semantics. Sarkar [17] identified five update operators, also based on [18]. These operators are: *Add, InsertAfter, Delete, Remove,* and *Replace*, and presented algorithms for the *Add* and *InsertAfter* operations. Zhan [22] proposed an extension to RQL, and defined a set of update operators. Both works define updates in an operational way, and semantic issues are considered to a very limited extent. Another approach was proposed by Ognyanov and Kiryakov [15]. The main statement of this approach is that the two basic types of updates in an RDF repository are the addition and the removal of a statement (triple). Then, the work turns simply into a description of a graph updating procedure, where labels indicate a version of the graph at a certain moment in time. Finally, Magiridou *et al* [14] recently proposed RUL, a declarative update language for RDF. They define three operations, *insert, delete* and *modify.* The proposal is based on RQL and RVL. The main drawback of this work is that it does not consider blank nodes and schema updates, i.e., the issues that raise the most interesting theoretical issues. Leaving these issues out turns the problem trivial. Thus, the authors basically end up dealing with changes to instances of classes.

3. PROBLEM STATEMENT

3.1 Review of Basic RDF Notions

We present here a streamlined version of RDF. The material of this subsection can be found in [5] with more detail.

There is an infinite set U (RDF URI references); an infinite set $B = \{N_j : j \in \mathbb{N}\}$ (Blank nodes); and an infinite set L (RDF literals). A triple $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple*. In such a triple, v_1 is called the *subject*, v_2 the *predicate* and v_3 the *object*. We often denote UBL the union of the sets U, B and L.

Definition 1. An *RDF graph* (just graph from now on) is a set of RDF triples. A *subgraph* is a subset of a graph. The *universe* of a graph G, universe(G), is the set of elements of UBL that occur in the triples of G. The *vocabulary* of G, denoted voc(G), is the set universe$(G) \cap (U \cup L)$. A graph is *ground* if it has no blank nodes. We also define the *union* of two graphs G_1, G_2, denoted $G_1 \cup G_2$, as the set theoretical union of their sets of triples.

RDFS Vocabulary. There is a set of reserved words defined in the RDF vocabulary description language, RDF Schema –just *rdfs-vocabulary* for us– that may be used to describe properties like attributes of resources (traditional attribute-value pairs), and also to represent relationships between resources. In this paper –following [5]– we will restrict to a fragment of this vocabulary which represents the essential features of RDF. It is constituted by the classes rdfs:Class [`class`] and rdf:Property [`prop`], and by the properties rdfs: range [`range`], rdfs:domain [`dom`], rdf:type [`type`], rdfs: subClassOf [`sc`] and rdfs:subPropertyOf [`sp`]. We present a semantics for this fragment, based on the following set of rules.

GROUP A (Subproperty)

$$\frac{(a, \texttt{type}, \texttt{prop})}{(a, \texttt{sp}, a)} \quad (1)$$

$$\frac{(a, \texttt{sp}, b) \;\; (b, \texttt{sp}, c)}{(a, \texttt{sp}, c)} \quad (2)$$

$$\frac{(a, \texttt{sp}, b) \;\; (x, a, y)}{(x, b, y)} \quad (3)$$

GROUP B (Subclass)

$$\frac{(a, \texttt{type}, \texttt{class})}{(a, \texttt{sc}, a)} \quad (4)$$

$$\frac{(a, \texttt{sc}, b) \;\; (b, \texttt{sc}, c)}{(a, \texttt{sc}, c)} \quad (5)$$

$$\frac{(a, \texttt{sc}, b) \;\; (x, \texttt{type}, a)}{(x, \texttt{type}, b)} \quad (6)$$

GROUP C (Typing)

$$\frac{(a, \texttt{dom}, c) \;\; (x, a, y)}{(x, \texttt{type}, c)} \quad (7)$$

$$\frac{(a, \texttt{range}, d) \;\; (x, a, y)}{(y, \texttt{type}, d)} \quad (8)$$

Definition 2 (Deductive System). Let G be a graph. For each rule $r : \frac{A}{B}$ above, define $G \vdash_r G \cup B$ iff $A \subseteq G$. Also define $G \vdash_s G'$ iff G' is a subgraph of G.

Define $G \vdash G'$ if there is a finite sequence of graphs $G_1, \ldots, G_n$ such that (1) $G = G_1$; (2) $G' = G_n$; and (3) for each i, either, $G_i \vdash_r G_{i+1}$ for some r, or $G_i \vdash_s G_{i+1}$.

Definition 3. Let G be an RDF graph. The *closure* of G, denoted $cl(G)$, is the maximal set of triples G' over universe(G) plus the rdfs vocabulary such that G' contains G and $G \vdash G'$.

In the next section we will need the logical notion of a *model* of a formula (an RDF graph). The model theory of RDF (given in [7]) follows standard classical treatment in logic with the notions of model, interpretation, and entailment (denoted $\models$). See [5] for details. Throughout this paper we will work with Herbrand models, which turn out to be special types of RDF graphs themselves. For a ground graph G, a Herbrand model of G is any RDF graph that contains cl(G) (in particular, cl(G) is a minimal model). From [5] the following results can be deduced.

Proposition 1. $G \models H$ *iff* $\mathrm{cl}(H) \subseteq \mathrm{cl}(G)$.

Theorem 1. *The deductive system of Definition 2 is sound and complete for* $\models$. *That is,* $G_1 \vdash G_2$ *iff* $G_1 \models G_2$.

3.2 The Problem

Consider the simplest problem related to the erase operation that we can find in RDF, and the associated semantic and complexity issues, namely: delete a tuple t or a set of tuples H, from an RDF graph G. To illustrate with a concrete example, let $G = \{(a, \texttt{sc}, b), (b, \texttt{sc}, c)\}$, and consider the following problems:

Problem 1: Erase $(a, \texttt{sc}, c)$ from G. Result: should $(a, \texttt{sc}, c)$ be derivable from G after the deletion?. If not, should we delete $(a, \texttt{sc}, b)$ or $(b, \texttt{sc}, c)$?

Problem 2: Erase $(a, \texttt{sc}, b)$ from G. Result: before deletion, $(a, \texttt{sc}, c)$ was implicit in G (it was entailed by G). Should it still be in G after deletion?. Should deletion be syntax-independent?

Problem 3: Erase $\{(a, \texttt{sc}, b), (b, \texttt{sc}, c)\}$ from G. Result: is it the empty set?. Either $(a, \texttt{sc}, b)$ or $(b, \texttt{sc}, c)$?. Again, should $(a, \texttt{sc}, c)$ be in the result?

A standard approach in KB is to ensure that, after deletion, the statement t should not derivable from G, and that the deletion should be minimal. The result should be expressed by another formula, usually in a more expressive language. For example, if in G above we erase $(a, \texttt{sc}, c)$, the "faithful" result should be something like $(a, \texttt{sc}, b) \vee (b, \texttt{sc}, c)$. But the problem is that we do not have disjunction in RDF.

In this paper we explore the behavior of the Katsuno-Mendelzon approach to define a semantics for update in RDF and concentrate on the characterization of the erase operation and its consequences over the formulas expressible in RDF. We will limit ourselves to study these questions for the case of ground graphs.

4. SEMANTICS OF UPDATE AND ERASE

In this section we address the problem introduced in Section 3.2. We characterize update and erase operations (i.e., adding or deleting an RDF graph H to/from another RDF graph G) using the Katsuno-Mendelzon approach, that is, identifying a theory with the set of models that satisfies it.

4.1 Katsuno-Mendelzon approach for RDF

The K-M approach to updates can be characterized as follows from a model-theoretic point of view: for each model M of the theory to be changed, find the set of models of the sentence to be inserted that are "closest" to M. The set of all models obtained in this way is the result of the change operation. Choosing an update operator then reduces to choosing a notion of closeness of models [4].

Definition 4. The operator $\circ$, representing the update of G with H, is defined as follows:

$$Mod(G \circ H) = \bigcup_{m \in Mod(G)} \min(Mod(H), \leq_m), \qquad (9)$$

where $\min(Mod(H), \leq_m)$ is the set of models of H minimal under $\leq_m$, which is a partial order depending on m.

We will use the following notion of distance between models, which gives us an order.

Definition 5 (Order). Let G, G_1, G_2 be models of RDF graphs with $voc(G) \subseteq voc(G_2), voc(G_1)$, and let $\mathcal{G}$ be a set of models of RDF graphs. The symmetric difference between two models G_1 and G_2, denoted as $G_1 \oplus G_2$, is $(G_1 \setminus G_2) \cup (G_2 \setminus G_1)$. Then : (1) define a relation $\leq_G$ such that $G_1 \leq_G G_2$ (G_1 is "closer" to G than G_2) if and only if $G_1 \oplus G \subseteq G_2 \oplus G$; (2) G_1 is $\leq_G$-minimal in $\mathcal{G}$ if G_1 is in $\mathcal{G}$, and if $G_2 \in \mathcal{G}$ and $G_2 \leq_G G_1$ then $G_2 = G_1$.

4.2 The notion of Update

Working with positive theories, the problem of update is fairly straightforward. The only concern is keeping the principle of irrelevance of syntax, i.e., the update should not depend on the particular syntax of the sentences involved.

Theorem 2. *Given the RDF graphs G and H, the update of G with H, $G \circ H$, is expressible as another RDF graph. Formally, $m \in (G + H)$ if and only if $m \in Mod(G \circ H)$.*

Proof. If $m \in Mod(G + H)$ then $m \in Mod(G)$ and $m \in Mod(H)$. Then $m_G = m$ is the model in $Mod(G)$ such that m is $\leq_{m_G}$-minimal in $Mod(H)$. Then, $m \in Mod(G \circ H)$. Conversely, let $m \in Mod(H)$ and $m_G \in Mod(G)$ such that m is $\leq_{m_G}$-minimal. Then $m_G \subseteq m$: otherwise, $(m \cup m_G) <_{m_G} m$, contradiction. Hence $m \models (G + H)$.

Proposition 2. *Let D, G, H be RDF graphs. Then, the definition of update satisfies the following statements: (1) $D \circ G \models G$; (2) if $D \models G$ then $D \circ G \equiv D$; (3) if $G_1 \equiv G_2$ and $H_1 \equiv H_2$ then $G_1 \circ H_1 \equiv G_2 \circ H_2$ (irrelevance of syntax); (4) $(D \circ G) + H \models D \circ (G + H)$; (5) if $D \circ G \models H$ and $D \circ H \models G$ then $D \circ G \equiv D \circ H$. (Note that these statements are an analogous, in our setting, of the K-M postulates for update not involving disjunction).*

4.3 The notion of Erase

Erasing statements from G means adding models to $Mod(G)$.

Definition 6. The operator $\bullet$, representing the erasure, is defined as follows: for graphs G and H, $G \bullet H$ is given by:

$$Mod(G \bullet H) = Mod(G) \cup \bigcup_{m \in Mod(G)} \min(((Mod(H))^c, \leq_m) \qquad (10)$$

and $(\ )^c$ denotes complement. In words, the models of $(G \bullet H)$ are those of G plus the collection of models $m_H \not\models H$ such that there is a model $m \models G$ for which m_H is $\leq_m$-minimal among the elements of $Mod(H)^c$. Compare identity (9).

Proposition 3. *Let D, G, H be RDF graphs. Then, the definition of erase satisfies the following statements: (1) $D \models D \bullet G$; (2) if $D \not\models G$ then $D \bullet G \equiv D$; (3) $D \bullet G \not\models G$; (4) if $G_1 \equiv G_2$ and $H_1 \equiv H_2$ then $G_1 \bullet H_1 \equiv G_2 \bullet H_2$; (5) $(D \bullet G) + G \models D$. (Note that these statements are an analogous, in our setting, of the K-M postulates for erase not involving disjunction).*

Representing faithfully in the RDF language the notions of update and erase defined above is not possible in the general case. The Update operator presents no difficulties, and it is in fact an RDF graph (formula). However, the Erase operator presents problems, arising from the fact that we have neither negation nor disjunction in RDF.

5. CHARACTERIZING DELETION IN RDF

The following notion is the key to obtain a workable characterization of erase (expressed previously only in terms of sets of models), based on the behavior over the formulas expressible in RDF.

Definition 7 (Erase Candidates). Let G and H be RDF graphs. Then the set of *erase candidates* of G and H, denoted $\mathrm{ecand}(G, H)$, is defined as the set of maximal subgraphs G' of $\mathrm{cl}(G)$ such that $G' \not\models H$.

Proposition 4. *Let G, H be models and $G \models H$. If $m \in (G - H)$, then there is a unique $E \in \mathrm{ecand}(G, H)$ with $m \models E$.*

Proof. Let $m \not\models H$ and $m_G \models G$ such that m is $\leq_{m_G}$-minimal. Assume $m_G \models \mathrm{cl}(G)$. Consider the subgraph $E = (m \cap m_G)$ of $\mathrm{cl}(G)$. Clearly $m \models E$, and hence and $E \not\models H$. *Claim:* $E \in \mathrm{ecand}(G, H)$. Assume $E \subseteq \mathrm{cl}(G)$ is not maximal with the property of not entailing H. Then there is $t \in (\mathrm{cl}(G) \setminus E)$ with $E \cup \{t\} \not\models H$. Then consider $m' = \mathrm{cl}(m \cup t)$. We have that $m' \not\models H$ and $m' <_{m_G} m$, a contradiction. The uniqueness of E follows from its maximality.

Theorem 3. *If $\mathrm{ecand}(G, H) = \{E\}$, then $(G \bullet H) \equiv E$.*

The theorem follows from the following proposition, that additionally states that $\mathrm{ecand}(G, H)$ defines a partition in the set of models defined by $G \bullet H$, and each such set is "represented" by the RDF graph E. Note that the smaller the size of $\mathrm{ecand}(G, H)$, the better the approximation to $G \bullet H$ of each element in $\mathrm{ecand}(G, H)$, being the limit Theorem 3.

We are ready for the theorem characterizing the RDF subgraph of $cl(G)$ which captures exactly all consequences of $G \bullet H$ expressible in RDF:

Theorem 4. *For all formulas F of RDF, $\bigcap \mathrm{ecand}(G, H) \models F$ if and only if $Mod(G \bullet H) \subseteq Mod(F)$.*

The proof follows from Proposition 4.

5.1 Computing Erase Candidates

From the discussion above, it follows the relevance of computing erase candidates to approximate $G \bullet H$. We will need the notion of proof sequence based on the deductive system from Section 3.

Definition 8 (Proof Sequence). Let G, H be RDF graphs. Then a *proof sequence* of H from G is a sequence of RDF graphs $H_1, \ldots, H_n$ such that:

1. $H_1 \subseteq G$ and $H \subseteq H_n$.
2. For each pair H_{i+1} and H_i one of the following holds:
   (a) (Standard rules) $H_{i+1} = H_i \cup \{t\}$, for $t_1, t_2 \in H_i$ and $\frac{t1 \;\; t_2}{t}$ is the instantiation of a rule (see rules in Secc 3).
   (b) (Mapping rule) $\mu(H_{i+1}) = H_i$ for a mapping μ.

Because of Theorem 1, proof sequences are sound and complete for testing entailment.

The first element in a proof sequence P will be called $\texttt{base}(P)$. $\texttt{base}(P)$ is a *minimal base* for the graphs G, H iff it is minimal under set inclusion among the bases of proofs of H from G, that is, for every proof P' of H from G, $\texttt{base}(P) \subseteq \texttt{base}(P')$. We refer to the set of minimal bases of G, H as $\texttt{minbases}(G, H)$.

We use the following notion of a cover for a collection of sets. A cover for a collection of sets $C_1, \ldots, C_n$ is a set C such that $C \cap C_i$ is non-empty for every C_i.

Lemma 1. *Let G, H be RDF graphs. C is a cover for the set $\texttt{minbases}(G, H)$ iff $(G \setminus C) \not\models H$.*

Proof. (If) If C is not a cover, then there is a minimal base $B \in (G \setminus C)$. Then there is a proof P for H from $G \setminus P$, where $\texttt{base}(P) = B$, contradicting that $(G \setminus C) \not\models H$. (Only If) Suppose not. Then there is a proof P for H from $G \setminus C$. We have that there is no minimal base B such that $B \subseteq \texttt{base}(P)$. Hence $\texttt{base}(P)$ is a minimal base for G, H, contradicting that C is a cover for all minimal bases.

Theorem 5. *Let G, H, D be RDF graphs. Then C is a minimal cover for the collection of sets $\texttt{minbases}(G, H)$ iff (i) $(G \setminus C) \not\models H$ and (ii) $G \setminus C$ is a maximal subgraph G' of G such that $G' \not\models H$.*

Proof. Follows from Lemma 1. It can be easily verified that the minimality of C implies the maximality of $G \setminus C$ and vice versa.

Corollary 1. *Let G, H, D be RDF graphs. $E \in \mathrm{ecand}(G, H)$ if and only if $E = \mathrm{cl}(G) \setminus C$ for C a minimal cover for the collection of sets $\texttt{minbases}(\mathrm{cl}(G), H)$.*

6. COMPLEXITY

In this section we study the complexity of computing an *erase* operation (computing *update* is straightforward). We show that computing erase candidates reduces to finding cuts in a class of directed graphs that encode RDF graphs.

Finding erase candidates reduces to compute RDF graphs we call *delta candidates.* We denote $\mathrm{dcand}(G, H)$ the set of RDF graphs $\{(\mathrm{nf}(G) \setminus G') : G' \in \mathrm{ecand}(G, H)\}$. Each of the graphs in $\mathrm{dcand}(G, H)$ will be called a *delta candidate* for G, H. Notice that the delta candidates can be alternatively defined as minimal graphs $D \subseteq \mathrm{cl}(G)$ such that $(\mathrm{cl}(G) \setminus D) \not\models H$.

6.1 Minimal Cuts

We will need the following standard notation related to cuts in graphs. Let (V, E) be a directed graph. A set of edges $C \subseteq E$ disconnects two vertices $u, v \subseteq V$ iff each path from u to v in the graph passes through a vertex in C. In this case C is called a *cut.* This cut is *minimal* if the removal of any node from C does not yield another cut. We also generalize cuts for sets of pairs of vertices yielding multicuts. A minimal multicut for a set of pairs of nodes $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ is a minimal set of edges that disconnects u_i and v_i. Given a graph G and a set of pairs of nodes N, we denote by $\texttt{MinCuts}(N, G)$ the set of minimal multicuts of N in G. Notice that when N has a single pair $\texttt{MinCuts}(N, G)$ is a set of cuts.

We will show that, in general, an element in $\mathrm{dcand}(G, H)$ is the union of two cuts: one defined in a directed graph we will denote $G[\texttt{sc}]$, and the other in a graph denoted $G[\texttt{sp}]$.

Given an RDF graph G, denote by $G[\texttt{sc}] = (N, V, \lambda)$ the labeled directed graph defined in Table 1 (above). For each triple of the form specified in the first column of the table, we have the corresponding edge in V. The set of nodes N consists of all the nodes mentioned in the edges given in the table. The function $\lambda : E \to G$ maps each edge in E to a triple in G, according to Table 1 (above). The labeled directed graph $G[\texttt{sp}]$ is defined similarly in Table 1 (below). As notation, we use the letters n and m to refer distinctly to nodes in $G[\texttt{sc}]$ and $G[\texttt{sp}]$, respectively.

Triple	Edge in $G[\texttt{sc}]$
$(a, \texttt{sc}, b)$	(n_a, n_b)
$(a, \texttt{type}, b)$	(n_a, n_b)
$(a, \texttt{type}, \texttt{class})$	(n_a, n_a)
	Edges in $G[\texttt{sp}]$
$(p, \texttt{sp}, q)$	(m_p, m_q)
(a, p, b)	$(m_{a,b}, m_p)$ $(m_{b,a}, m_p)$
$(p, \texttt{dom}, c)$	$(m_p, m_{c,\texttt{dom}})$
$(p, \texttt{range}, c)$	$(m_p, m_{c,\texttt{range}})$

Table 1: Description of the graphs $G[\texttt{sc}]$ (above) and $G[\texttt{sp}]$ (below).

For an RDF triple t we define a set of pairs of nodes that specified the cut problems related to the erase of the triple t from an RDF graph G. The set $t[\texttt{sc}, G]$ will contain pairs of nodes in the graph $G[\texttt{sc}]$ and the set $t[\texttt{sp}, G]$ will contain pairs of nodes in $G[\texttt{sp}]$. Formally, we denote by $t[\texttt{sc}, G]$ the pairs of nodes (u, v), u, v nodes in $G[\texttt{sc}]$ as described in Table 2 (second column). Analogously, we define $t[\texttt{sp}, G]$ using Table 2 (third column). As an example, for a triple of the form $(a, \texttt{sc}, b)$ in a graph G, $(a, b, c)[\texttt{sc}, G]$ contains the single pair of nodes (n_a, n_b), where both nodes n_a, n_b belong to $G[\texttt{sc}]$. Notice that there is always a single pair of nodes in $t[\texttt{sc}, G]$, and the only case where $t[\texttt{sc}, G]$ may have several pairs of nodes is when t is of the form $(a, \texttt{type}, b)$.

For an RDF graph U, $U[\texttt{sc}, G]$ is the union of the sets $t_i[\texttt{sc}, G]$, for the triples t_i in U.

6.2 Complexity of Erase

For the sake of space, we will present here the case where the graph to erase has a single triple. Our results can be easily generalized to computing erase candidates $\mathrm{ecand}(G, H)$ for the case where H has several triples.

A delta $\mathrm{dcand}(G, t)$, will be defined with two sets of graphs, denoted $\mathrm{dcand}_{\texttt{sc}}(G, t)$ and $\mathrm{dcand}_{\texttt{sp}}(G, t)$. For each $D \in \mathrm{dcand}(G, t)$, $D = D_1 \cup D_2$, for of any two RDF graphs

Triple $t \in G$	$t[\mathtt{sc}, G]$	$t[\mathtt{sp}, G]$
$(a, \mathtt{sc}, b)$	(n_a, n_b)	–
$(a, \mathtt{sp}, b)$	–	(m_a, m_b)
(a, p, b)	–	(m_{ab}, m_p)
$(a, \mathtt{type}, c)$	(n_a, n_c)	pairs $(m_{a,x}, m_{c,\mathtt{dom}})$ for all x
		pairs $(m_{x,a}, m_{c,\mathtt{range}})$ for all x

Table 2: Pair of nodes $t[\mathtt{sc}, G]$ and $t[\mathtt{sp}, G]$ associated to a triple t in a graph G.

$D_1 \in \mathrm{dcand}_{\mathtt{sc}}(G, t)$ and $D_2 \in \mathrm{dcand}_{\mathtt{sp}}(G, t)$.

Proposition 5. *Let G be an RDF graph, $G' = \mathrm{cl}(G)$, and consider a triple t. The following holds: (i)* $\mathrm{dcand}_{\mathtt{sc}}(G, t) = \mathtt{MinCuts}(G'[\mathtt{sc}], t[\mathtt{sc}, G'])$*; (ii)* $\mathrm{dcand}_{\mathtt{sp}}(G, t) = \mathtt{MinCuts}(G'[\mathtt{sp}], t[\mathtt{sp}, G'])$*.*

PROOF. (Sketch) Corollary 1 can be expressed in terms of delta candidates as follows. Let G, H, D be RDF graphs. Then $D \in \mathrm{dcand}(G, H)$ iff D is a minimal cover set for $\mathtt{minbases}(\mathrm{cl}(G), H)$.

We sketch the proof for the case where t is of the form $(a, \mathtt{sc}, b)$. In this case we can verify that $\mathtt{minbases}(G', H)$ corresponds to the RDF triples associated to the simple paths (paths with no cycles) from n_a to n_b in $G[\mathtt{sc}]$. Therefore, it follows that the minimal cuts $\mathtt{MinCuts}(G'[\mathtt{sc}], t[\mathtt{sc}, G']$ are exactly the delete candidates $\mathrm{dcand}(G, t)$. Notice that in the case where t is of the form $(a, \mathtt{sc}, b)$, $\mathrm{dcand}(G, t) = \mathrm{dcand}_{\mathtt{sc}}(G, t)$, because, in this case $\mathrm{dcand}_{\mathtt{sp}}(G, t)$ is empty.

Theorem 6. *Let G, H be ground RDF graphs, and t be a ground a triple. The problem of deciding whether $E \in$* $\mathrm{ecand}(G, t)$ *is in PTIME.*

PROOF. (Sketch) From Proposition 5, the problem reduces to determine if $D = \mathrm{cl}(G) \setminus E$ is a delta candidate in $\mathrm{dcand}(G, t)$. Let $G' = \mathrm{cl}(G)$, G' can be computed in polytime. The triples in D yield two sets of edges $\mathrm{dcand}_{\mathtt{sc}}$ and $\mathrm{dcand}_{\mathtt{sp}}$ in the graphs $G'[\mathtt{sc}]$ and $G'[\mathtt{sp}]$, respectively. Thus we have to test (i) whether $t[\mathtt{sc}, G']$ is a minimal cut in $G'[\mathtt{sc}]$ and (ii) whether $t[\mathtt{sp}, G']$ is a minimal (multi)cut in $G'[\mathtt{sp}]$. In both cases the test can be done in PTIME by simple reachability analysis in the graphs $G'[\mathtt{sc}]$ and $G'[\mathtt{sp}]$, respectively. Testing whether a set of edges S is a minimal cut for (v_1, u_1) in a graph $GR = (V, E)$ can be done by performing polytime reachability analysis in the graph as follows. To test whether S is a cut, delete from E the edges in S, and test whether v_1 reaches u_1 in this new graph. To test minimality, do the same test for each set of edges $S' \subset S$ resulting from removing a single edge from S. S is minimal iff all of the S's are not cuts. We proceed similarly for testing if a set of edges is a minimal multicut.

7. CONCLUSIONS

In this paper we considered an RDF database as a knowledge base, and treated the problem of updating the database in the framework of the traditional proposals of knowledge base updating. We characterized the update of a graph G with a graph H within the framework of the K-M approach, and defined the meaning of the *update* and *erase* operations in RDF on a solid foundation (considering, in the latter case, that we do not have negation nor disjunction in RDF). We also provided algorithms for calculating these operations, including a detailed complexity analysis. In future work we will develop an update language for RDF, and extend our study to more expressive languages, like OWL.

8. REFERENCES

[1] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *International Conference on Very Large Databases(VLDB'85)*, Stockholm, Sweden, 1985.

[2] R. Angles and C. Gutierrez. Querying RDF data from a graph database perspective. In *European Conference on the Semantic Web (ECSW'05)*, pages 346–360, 2005.

[3] P. Gardenförs. Conditionals and changes of belief. *Acta Philosophica Fennica, Vol. XX*, pages 381–404, 1978.

[4] G. Grahne, A.O. Mendelzon, and P. Z. Revesz. Knowledgebase transformations. *Journal of Computer and System Sciences, Vol 54(1)*, pages 98–112, 1997.

[5] C. Gutierrez, C. Hurtado, and A.O. Mendelzon. Foundations of semantic web databases. In *23rd. Symposium on Principles of Database Systems*, pages 95–106, 2004.

[6] C. Gutierrez, C. Hurtado, and A. Vaisman. Temporal RDF. In *European Conference on the Semantic Web (ECSW'05) (Best paper award)*, pages 93–107, 2005.

[7] Patrick Hayes(Ed.). RDF semantics. *W3C Working Draft, October 1st., 2003.*

[8] A.J.H. Hidders. A graph-based update language for object-oriented data models. *Doctoral Thesis, Technische Universiteit Eindhoven, The Netherlands*, 2001.

[9] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of ACM, 31(4)*, pages 761–791, 1984.

[10] H Katsuno and A. O. Mendelzon. On the difference between updating knowledge base and revising it. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 387–394, Cambridge, MA, 1991.

[11] A.M. Keller and M. Winslett. On the use of extended relational model to handle changing incomplete information. *IEEE Trans. on Software Engineering, SE-11:7*, pages 620–633, 1985.

[12] O. Lassila and R.(Eds.) Swick. Resource description framework (RDF) model and syntax specification. *W3C Working Draft, 1998.*

[13] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. Establishing the semantic web 11: An infrastructure for searching, reusing, and evolving distributed ontologies. In *International Conference on World Wide Web*, pages 439–448, 2003.

[14] M. Magiridou, S. Sahtouris, S. Christophides, and M Koubarakis. RUL: A declarative update language for RDF. In *International Semantic Web Conference*, pages 506–521, 2005.

[15] D. Ognyanov and A. Kiryakov. Tracking changes in rdf(s) repositories. In *EKAW'02*, pages 373–378, Spain, 2002.

[16] E. Prud'Hommeaux and A. Seaborne (Eds.). SPARQL query language for rdf. *W3C Working Draft, July, 2005.*

[17] S. Sarkar and H.C. Ellis. Five update operations for rdf. *Rensselaer at Hartford Technical Report, RH-DOES-TR 03-04*, 2003.

[18] I. Tatarinov, G. Ives, A. Halevy, and D. Weld. Updating XML. In *Proceedings of ACM SIGMOD Conference*, pages 413–424, Santa Barbara, California, 2001.

[19] U. Visser. Intelligent information integration for the semantic web. *Lecture Notes in Artificial Intelligence (3159)*, 2004.

[20] World Wide Web Consortium. *RDF semantics*, 2004. http://www.w3.org/TR/rdf-mt.

[21] World Wide Web Consortium. *XQuery Update Facility Requirements (working draft)*, 2005. http://www.w3.org/TR/2005/WD-xquery-update-requirements-20050603/.

[22] Y. Zhan. Updating RDF. In *21st Computer Science Conference*, Rensselaer at Hartford, 2005.