Updating Semantic Web Data

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

ABSTRACT

The basic data model for the Semantic Web is the Resource Description Framework (RDF). From a data management point of view, it resembles a lightweight knowledge base. 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. Viewing RDF as a knowledge base, and using the formalism of Katsuno-Mendelzon, we define semantics for updates in RDF. Then we show that RDF as a representation system does not have enough expressiveness to state updates satisfying the Katzuno-Mendelzon framework. Hence, we study the behavior of classical update operations in the framework of RDF, and propose versions of such updates based on approximations. Finally, we study how to compute these operations, which for erase are particularly complex, and give complexity bounds for them.

1. INTRODUCTION

The *Semantic Web* is a proposal oriented to represent Web content in an easily machine-processable way [30, 3]. 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) [20]. From a conceptual point of view, RDF resembles a fragment of the binary first-order logic, including features like transitivity of some predicates. 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 the RDF model, the universe to be modeled is a set of *resources* (anything that can have a universal identifier, URI), described by a language consisting basically in a set of *properties*, technically binary predicates. Descriptions are *statements* of the form subject-predicate-object. Subjects and objects can be anonymous elements, called *blank nodes*. The RDF specification also includes a built-in vocabulary, namely (RDFS) [6], dealing essentially with typing and inheritance of classes and properties. The natural interconnection produced by a set of RDF statements over a given set resources makes that RDF data closely resemble labeled graphs. A natural problem that arises in this context, due to the huge volumes of data involved, is the management of RDF data. In fact, in the last years we have seen increasing activity in formalizing issues related to RDF data management and in developing tools to process such data.

RDF allows describing the concepts and relationships that may exist in a community of people and software agents. Thus, RDF data is subject to changes. Some studies have addressed changes in an ontology [22, 29]. More recently, the representation and querying temporal information in RDF [13] has been also studied. In this paper we concentrate on the important 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 [24, 27, 33, 23], and tackled the subject from a syntactical point of view.

We will illustrate with an example the importance of accounting for the changes that can occur throughout the life cycle of the data and the problems that may appear when there is RDF vocabulary involved. Let us consider a Web music store. We wish to use Semantic Web technology in order to make it easier for the user to find information about artists depending on the music style they are looking for. Thus, we will define an ontology using RDF in order to describe the components of the site. Figure 1 shows an RDF representation of a portion of an ontology designed for a music web site. There are three music styles (*blues, rock* and *jazz*), defined as subclasses of *genre*. Each of these styles has in turn other styles as subclasses. We have also included an ontology for artists with properties (e.g. *performs*), and subproperties (e.g. *plays guitar*). A music web site is usually a very dynamic environment. For example, if at a certain point in time a new artist is added, say, *Bee Gees*, a triple containing the new artist is added (*Bee Gees*, `type`, *disco*). Other changes can be the addition of new artists, music styles, relationships between objects, and so on. A possible new scenario is depicted in Figure 2, where changes are indicated in dashed lines.

1.1 Problem Statement

Updates and Revision. The semantics of updates for data models becomes difficult when the model turns –even slightly–

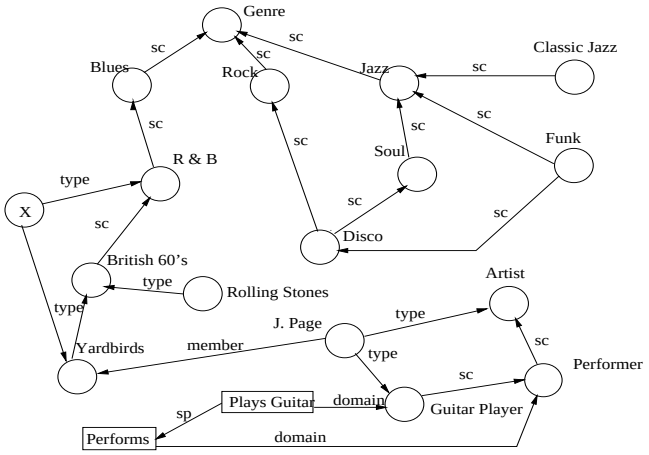

Figure 1: RDF graph for a music web site. Label sc indicates subclass, and sp subproperty.

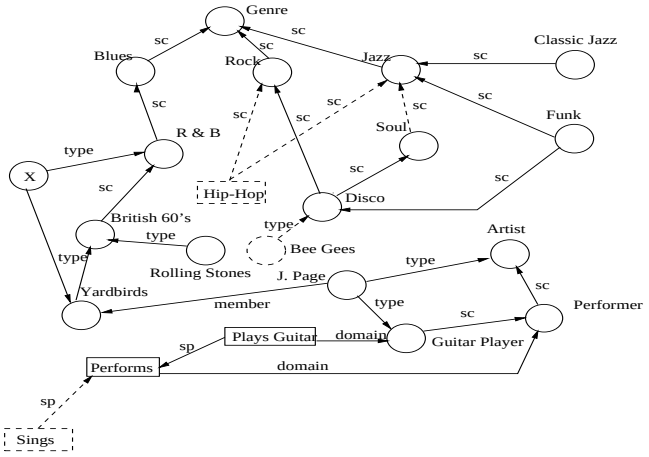

Figure 2: The music web site after some changes.

more general than a simple relational structure [11]. 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 [18] 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* [19, 18]. *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. The RDF model is more than a simple relational structure; its expressivity resembles features of the existential conjunctive fragment of first order logic plus transitivity of some predicates and inheritance axioms. 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 [31]. By some classic results of Gardenförs [9], the notion of revision becomes trivial in this setting. On the other hand, when viewing RDF from a database point of view, the notion of update becomes relevant. In this paper we concentrate in this latter notion, and follow the approach of Katsuno and Mendelzon [18], which we considered the best suited for RDF. In the last section of the paper we discuss other possible approaches and compare them with our work. The question that arise is under which conditions we can express the notion of the ideal update operation of Katzuno-Mendelzon in RDF, and when this is not possible, what is a good approximation to such operation.

Updating RDF. Management, in particular maintainability, of RDF data needs a well defined notion of update. The problem becomes particularly relevant since the standardization of a query language for RDF [25]. We will show that the problem of characterizing these changes in RDF is far from being trivial and raises interesting practical and theoretical issues. Consider for example the problem of deleting all triples containing the value *artist* in Figure 3 (a). The result, clearly, is the one shown in Figure 3 (b), where dashed lines indicate 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 4 (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 $(guitar\ player, \texttt{sc}, artist)$? In the former case, it should stay; in the latter it should be deleted. What is to be done? This paper studies these and other issues regarding update semantics in RDF.

1.2 Contributions

In this paper, we study the problem of updating RDF data (which seems to have been overlooked by the RDF community), treating the problem in the framework of updating knowledge bases, but also considering the limited expressiveness of RDF.

We introduce a sound semantics for RDF update and erase operations based on solid grounds, using a model-theoretic characterization based on the Katsuno-Mendelzon approach.

We investigate RDF as a representation system, concluding that it cannot express the erase operation as classically defined. To overcome this limitation, we define a notion of erase expressible in RDF, which is an approximation of the standard erase operator for knowledge bases. In particular, we prove that this approximation is the best approximation expressible in RDF for the notion of erase given in the framework Katsuno-Mendelzon.

We give operational procedures for calculating erase operations, based on simple proof-theoretic notions of RDF. We investigate the theoretical complexity issues associated to the operations of update and erase in RDF. In particular, we study the connection of the erase problem with the problem of finding minimal multicuts in directed graphs. This connection leads us to prove complexity bounds and characterize well-behaved subclasses.

The remainder of the paper is organized as follows. In Section 2 we review RDF concepts and present a formalization of RDF apt to develop the subsequent logical and mathemat-

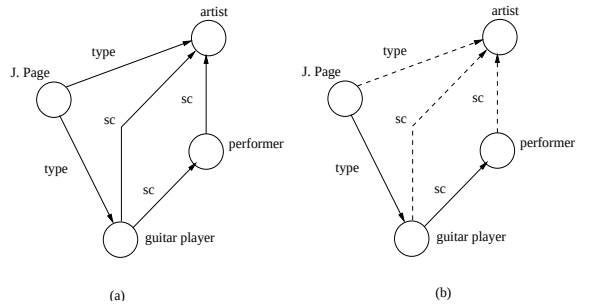

Figure 3: Deleting all triples containing "artist".

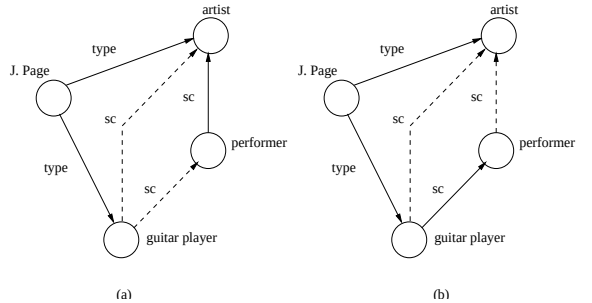

Figure 4: Deleting the triple $(guitarplayer, \texttt{sc}, artist)$**.**

ical framework. In Section 3 we introduce and discuss our semantics for updates in RDF and the necessary background on the Katsuno-Mendelzon approach. Section 4 presents our proposal to overcome the limitations of RDF to express the erase operator. Section 5 studies the complexity of the update and erase operations proposed. Finally, in Section 6 we give a thorough account of related work in the field and their relation with our approach. We conclude in Section 7.

2. PRELIMINARIES

2.1 An abstract formalization of RDF

We present here a streamlined version of RDF. The material of this subsection can be found in [12] with more detail.

Assume 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 will use letters $N, X, Y, \ldots$ to denote blank nodes, and $a, b, c, \ldots$ for URIs and literals. Graphically we represent RDF graphs as follows: each triple (a, b, c) is represented by $a \stackrel{b}{\rightarrow} c$. Note that the set of arc labels can have non-empty intersection with the set of node labels.

We will need some technical definitions. A *map* is a function μ : UBL $\rightarrow$ UBL preserving URIs and literals, i.e., $\mu(u) = u$ and $\mu(l) = l$ for all $u \in U$ and $l \in L$. A map is *ground* if its image does not intersect B. Given a graph G, we define $\mu(G)$ as the set of all $(\mu(s), \mu(p), \mu(o))$ such that $(s, p, o) \in G$. A map μ is *consistent* with G if $\mu(G)$ is an RDF graph, *i.e.* , if s is the subject of a triple, then $\mu(s) \in UB$, and if p is the predicate of a triple, then $\mu(p) \in U$. In this case, we say that the graph $\mu(G)$ is an *instance* of the graph G. An instance of G is *proper* if $\mu(G)$ has fewer blank nodes than G. This means that either μ sends a blank node to an URI or a literal, or identifies two blank nodes of G. We will overload the meaning of map and speak of a *map* $\mu : G_1 \rightarrow G_2$ if there is a map μ such that $\mu(G_1)$ is a subgraph of G_2.

Two graphs G_1, G_2 are *isomorphic*, denoted $G_1 \cong G_2$, if there are maps μ_1, μ_2 such that $\mu_1(G_1) = G_2$ and $\mu_2(G_2) = G_1$.

We define two operations on graphs. The *union* of G_1, G_2, denoted $G_1 \cup G_2$, is the set theoretical union of their sets of triples. The *merge* of G_1, G_2, denoted $G_1 + G_2$, is the union $G_1 \cup G_2'$, where G_2' is an isomorphic copy of G_2 whose set of blank nodes is disjoint with that of G_1. Note that $G_1 + G_2$ is unique up to isomorphism.

RDFS Vocabulary. There is a set of reserved words defined in the RDF vocabulary description language, RDF Schema [6], –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. It defines classes and properties that may be used for describing groups of related resources and relationships between resources.

In this paper –following [12]– 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`].

2.2 A deductive system for RDF

We present a semantics for our fragment based on a set of rules. The set of rules is arranged in four groups. Group A describes the semantics of blank nodes, which is essentially the semantics of RDF graphs without rdfs vocabulary. Group B describes the semantics of `sp`, stating that it is a transitive relation. Group C describes similar semantics for `sc`. Group D states the semantics of `dom` and `range`, the domain and range of a relation.

GROUP A (Existential) For a map $\mu : G' \rightarrow G$:

$$\frac{G}{G'}$$

GROUP B (Subproperty)

$$\frac{(a, \texttt{type}, \texttt{prop})}{(a, \texttt{sp}, a)} \qquad \frac{(a, \texttt{sp}, b)\ \ (b, \texttt{sp}, c)}{(a, \texttt{sp}, c)}$$

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

GROUP C (Subclass)

$$\frac{(a, \texttt{type}, \texttt{class})}{(a, \texttt{sc}, a)} \qquad \frac{(a, \texttt{sc}, b)\ \ (b, \texttt{sc}, c)}{(a, \texttt{sc}, c)}$$

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

GROUP D (Typing)

$$\frac{(a, \texttt{dom}, c)\ \ (x, a, y)}{(x, \texttt{type}, c)} \qquad \frac{(a, \texttt{range}, d)\ \ (x, a, y)}{(y, \texttt{type}, d)}$$

The following deductive system is based on standard rules defined in [15].

Definition 2 (Deductive System). Let G be a graph. For each rule $r : \frac{A}{B}$ above, define $G \vdash_r G \cup \mu(B)$ iff there is a map $\mu : A \to G$. 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 $G_i \vdash_s G_{i+1}$.

2.3 Semantics and model theory of RDF

The model theory of RDF (given in [15]) follows standard classical treatment in logic with the notions of model, interpretation, entailment, etc. plus some non-standard features arising from the fact that predicates can also be elements of the universe.

Throughout this paper we will work with Herbrand models. On these lines, the models of RDF are special types of RDF graphs themselves.

Definition 3. A (Herbrand) *model* m of G is a ground RDF graph m closed under $\vdash$ such that there is a map μ with $\mu(G) \subseteq m$. We denote this as $m \models H$. Usually if the map is known, we will write $m \models G[\mu]$.

We define an *entailment* relationship between RDF graphs as follows: $G \models H$ iff for every model m, if $m \models G$ then $m \models H$.

Two graphs are *equivalent*, denoted $G_1 \equiv G_2$, if $G_1 \models G_2$ and $G_2 \models G_1$. Note that isomorphism is a purely syntactic relation among graphs, but equivalence relies on the semantic notion of entailment.

The following theorem establishes the relation between the deductive system and the entailment relation. The proof follows the same lines of similar results in [15, 12] with slight modifications.

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

Definition 4. Let G be an RDF graph.
1. G is *lean* if there is no map μ such that $\mu(G)$ is a proper subgraph of G.
2. A *closure* of G is a maximal set of triples G' over universe(G) plus the rdfs vocabulary such that G' contains G and is equivalent to it.
3. The *normal form* of G, denoted $\mathrm{nf}(G)$, is defined as $\mathrm{lean}(G')$ for a closure G' of G.

It turns out that although there could be several different closures for a graph G, its normal form is unique [12].

Theorem 2 (cf. [15, 12]). $G_1 \models G_2$ *if and only if there is a map from* G_2 *to* $\mathrm{cl}(G_1)$.

Minimal models also will play a role in what follows. The *minimal model checking problem for RDF* is the following problem: Given an RDF graph G and a ground assignment μ, is $\mathrm{cl}(\mu(G))$ a minimal model for G?

Proposition 1. *1. Let* $G \models_{min} H$ *iff for all minimal models* m *of* G, $m \models H$. *Then* $G \models_{min} H$ *iff* $G \models H$.
2. The minimal model checking problem for RDF is co-NP complete.
3. The minimal model checking problem for ground RDF graphs can be solved in polynomial time.

3. SEMANTICS OF UPDATE AND ERASE

3.1 An example as motivation

Consider the simplest problem related to the erase operation that we can find in RDF, and the semantic issues and complexity associated to it:

Problem: Delete the tuple t from the graph G.

To be more concrete, let $G = \{(a, \texttt{sc}, b), (b, \texttt{sc}, c), (X, \texttt{sc}, c)\}$, 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. And $(X, \texttt{sc}, c)$?.

Problem 4: Erase $\{(X, \texttt{sc}, c)\}$ from G. Result: G again?. Or $\{(a, \texttt{sc}, b)\}$?.

Some complexity issues. 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 decision problem: "Is G' a subgraph of G maximal such that t is not entailed?" is highly complex, more precisely DP-complete. Nevertheless, the previous problem can be solved in polynomial time if we keep graphs in normal form, i.e., instead of G we keep $\mathrm{nf}(G)$. But, calculating normal forms is as complex as the above problem. Another source of complexity is that the number of the "candidates" for erase can be exponential, even when deleting only one tuple.

Language Expressiveness. In general, when deleting parts of the graph G to avoid entailment of t, one has to delete more than it would be strictly necessary. In general, the result should be expressed by another formula. For example, if in G above we erase $(a, \texttt{sc}, c)$, the "faithful" result should be something like $(a, \texttt{sc}, b) \vee (b, \texttt{sc}, c)$. The problem is that we do not have disjunction in RDF.

3.2 Katsuno-Mendelzon approach

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 theory that describes 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 [11].

Definition 5. The update of G with H is the logical expression whose models are

$$Mod(G \circ H) = \bigcup_{m \in Mod(G)} \min(Mod(H), \leq_m) \qquad (1)$$

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 6 (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.

1. The symmetric difference between two models G_1 and G_2, denoted as $G_1 \oplus G_2$, is $(G_1 \setminus G_2) \cup (G_2 \setminus G_1)$.
2. 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 $G_2 \oplus G \models G_1 \oplus G$. (This notion is a little bit weaker than the traditional $G_1 \oplus G \subseteq G_2 \oplus G$.)
3. 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 $G_2 = G_1$.

It is not difficult to check that $\leq_G$ is a partial order.

3.3 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.

Definition 7 (Update). Let G be a database and H a graph. Define the update of G with H, $G \circ H$, as $G + H$ (recall that $+$ is the merge operation).

Theorem 3. *The update operation defined above satisfies the K-M approach, that is: $m \in Mod(G \circ H)$ if and only if $m \in Mod(H)$ and there exists a model $m_G \in Mod(G)$ such that m is $\leq_{m_G}$-minimal.*

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 $Mod(H)$. 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, the fragment of the K-M postulates for update not involving negation nor disjunction:*

1. $D \circ G \models G$,
2. *If* $D \models G$ *then* $D \circ G \equiv D$,
3. *(Irrelevance of syntax) If* $G_1 \equiv G_2$ *and* $H_1 \equiv H_2$ *then* $G_1 \circ H_1 \equiv G_2 \circ H_2$,
4. $(D \circ G) + H \models D \circ (G + H)$,
5. *If* $D \circ G \models H$ *and* $D \circ H \models G$ *then* $D \circ G \equiv D \circ H$.

3.4 The notion of Erase

Erasing a statement from G means adding models to $Mod(G)$. In the original K-M approach, G erase H, denoted $G \bullet H$, is defined as $G \vee (G \circ \neg H)$, where $\circ$ is the update operator [18]. Another way of formulating this statement in terms of models is:

Lemma 1. *The erase of $G \bullet H$ is the logical sentence whose set of models is $Mod(G) \cup (G - H)$, where*

$$G - H = \bigcup_{m \in Mod(G)} \min((Mod(H))^c, \leq_m) \qquad (2)$$

and $(\ )^c$ denotes complement. In words, $(G - H)$ is 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 (1).

Proposition 3. *Let D, G, H be RDF graphs. Then the definition of erase satisfies the following statements, the fragment of the K-M postulates for erase not involving negation nor disjunction:*

1. $D \models D \bullet G$,
2. *If* $D \not\models H$ *then* $D \bullet G \equiv D$,
3. $D \bullet G \not\models G$,
4. *(Irrelevance of Syntax) 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. $(D \bullet G) + G \models D$.

4. A PROPOSAL FOR RDF

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. The issue of expressiveness of *representation systems* when dealing with update has been extensively studied for several systems (see Section 6.1). To the best of our knowledge, there is no such work for RDF.

A solution to the above limitation is to express the Erase by means of the best possible approximation using the expressiveness of the RDF language. The best scenario is when $G \bullet H$ is expressible in RDF. It turns out that this happens rarely. In fact, when $|\,\mathrm{nf}(H)| \geq 2$, there is no RDF formula equivalent to $G \bullet H$, unless G is trivial. Even if $|\,\mathrm{nf}(H)| = 1$, there are few favorable cases (see Theorem 4 below).

If $G \bullet H$ is not expressible in RDF, then we need to approximate it. The following section deals with this approach.

4.1 Approximating the Erase Operator

The approach we are going to follow can be delineated as *minimalist*, in the sense that we eliminate the *minimum* information necessary to satisfy the user request.

Definition 8 (Erase Candidates). Let G and H be RDF graphs. Then the set of *erase candidates* of G and H, denoted $\text{ecand}(G, H)$, is defined as the set of maximal subgraphs G' of $\text{nf}(G)$ such that $G' \not\models H$.

Theorem 4. *If* $\text{ecand}(G, H) = \{E\}$*, then* $(G \bullet H) \equiv E$.

The theorem follows from the following proposition, that additionally states that $\text{ecand}(G, H)$ 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 $\text{ecand}(G, H)$, the better the approximation to $G \bullet H$ of each element in $\text{ecand}(G, H)$, being the limit Theorem 4.

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

Proof. Let $m \not\models H$ and $m_G \models G$ such that m is $\leq_{m_G}$-minimal. Assume $m_G \models \text{nf}(G)[\mu]$. Consider the subgraph $E = \mu^{-1}(m \cap m_G)$ of $\text{nf}(G)$. Clearly $m \models E$ (via μ) and hence and $E \not\models H$. *Claim:* $E \in \text{ecand}(G, H)$. Assume $E \subseteq \text{nf}(G)$ is not maximal with the property of not entailing H. Then there is $t \in (\text{nf}(G) \setminus E)$ with $E \cup \{t\} \not\models H$. Then consider $m' = \text{cl}(m \cup \mu(t))$. We have that $m' \not\models H$ and $m' <_{m_G} m$, a contradiction. The uniqueness of E follows from its maximality.

Next, we will prove that the set of erase candidates is the best approximation when considering sets of RDF graphs as formulas for $Mod(G \bullet H)$. In fact, there is no set of RDF graphs whose set of models is closer to the set of models of $G \bullet H$ than $\text{ecand}(G, H)$:

Theorem 5. *There is no set* S *of RDF graphs such that* $Mod(G \bullet H) \subseteq Mod(S) \subset Mod(ecand(G, H))$.

Proof. First, from Proposition 4 and an easy check follows that $Mod(G \bullet H) \subseteq Mod(\text{ecand}(G, H)$. So, assume there is a set $S = \{G_1, ...G_2\}$ as in the statement of the theorem. Then there is at least one $E \in \text{ecand}(G, H)$ such that there is no map from any G_i to $\text{nf}(E)$. Now consider the free model e of E, that is the model of $nf(E)$ which replaces every blank X by a fresh constant c_X. Hence $e \notin Model(S)$. *Claim:* $e \in (G - H)$. First $e \in Mod^c(H)$, otherwise $E \models H$, a contradiction. Now, consider the free model g of G such that each variable X is map to the same constant c_X to which X was mapped in e. From the maximality of E follows that e is $\leq_g$-minimal in $Mod^c(H)$. Hence $e \in G - H \subseteq Mod(S)$, contradiction.

4.2 Computing Erase Candidates

From the discussion in the previous section, 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 Secc 2.2.

Definition 9 (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 it holds one of the following:
   (a) (Standard rules) $H_{i+1} = H_i \cup \{t\}$, for $t_1, t_2 \in H_i$ and $\frac{t1 \;\; t2}{t}$ is the instantiation of a rule (see rules in Secc 2.2).
   (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 2. *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 6. *Let* G, H, D *be RDF graphs. Then* C *is a minimal cover for the collection of sets* $\texttt{minbases}(G, H)$ *iff (i)* $(C \setminus C) \not\models H$ *and (ii)* $G \setminus C$ *is a maximal subgraph* G' *of* G *such that* $G' \not\models H$.

Proof. Follows from Lemma 2. 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.* $E \in \text{ecand}(G, H)$ *if and only if* $E = \text{nf}(G) \setminus C$ *for* C *a minimal cover for the collection of sets* $\texttt{minbases}(\text{nf}(G), H)$.

5. COMPLEXITY

In this section we study the complexity of computing an erase operation (computing update operations is straightforward). We show that computing erase candidates reduces to finding cuts in a class of directed graphs that encode RDF graphs. In Section 5.1 we explain such directed graphs. In Section 5.2 we study the complexity of erase operations for ground graphs. In Section 5.3 we present the complexity for the general case.

The problem of finding erase candidates reduces to computing RDF graph we call *delta candidates*. We introduce the

notion of delta candidates because delta candidates correspond to minimal cuts, which makes the presentation in this section simpler. We denote $\text{dcand}(G, H)$ the set of RDF graphs $\{(\text{nf}(G) \setminus G') \ : \ G' \in \text{ecand}(G, H)\}$. Each of the graphs in $\text{dcand}(G, H)$ will be called a *delta candidate* for G, H. Notice that the delete candidates can be alternatively defined as minimal graphs $D \subseteq \text{nf}(G)$ such that $(\text{nf}(G) \setminus D) \not\models H$.

5.1 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, a delta candidate in $\text{dcand}(G, H)$ is the union of two cuts. One is defined in a directed graph we will denote as $G[\texttt{sc}]$, 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.

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

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

5.2 Ground Graphs

We first study the case where the graph to erase has a single triple. Then we study erase operations for ground graphs with several triples.

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

Proposition 5. *Let G be an RDF graph, $G' = \text{nf}(G)$, and consider a triple t. Then the following holds:*
(i) $\text{dcand}_{\texttt{sc}}(G, t) = \texttt{MinCuts}(G'[\texttt{sc}], t[\texttt{sc}, G'])$.
(ii) $\text{dcand}_{\texttt{sp}}(G, t) = \texttt{MinCuts}(G'[\texttt{sp}], t[\texttt{sp}, G'])$.

Proof. (Sketch) First we express Corollary 1 in terms of delta candidates as follows. Let G, H, D be RDF graphs. Then $D \in \text{dcand}(G, H)$ iff D is a minimal cover set for $\texttt{minbases}(\text{nf}(G), H)$.

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

Theorem 7. *Let G, H be ground RDF graphs, and t be a ground a triple. The problem of deciding whether $E \in \text{ecand}(G, t)$ is in PTIME.*

Proof. (Sketch) From Proposition 5 it follows that the problem reduces to determine if $D = \text{nf}(G) \setminus E$ is a delta candidate in $\text{dcand}(G, t)$. Let $G' = \text{nf}(G)$, G' can be computed in polytime. The triples in D yield two sets of edges $\text{dcand}_{\texttt{sc}}$ and $\text{dcand}_{\texttt{sp}}$ in the graphs $G'[\texttt{sc}]$ and $G'[\texttt{sp}]$, respectively. Thus we have to test (i) whether $t[\texttt{sc}, G']$ is a minimal cut in $G'[\texttt{sc}]$ and (ii) whether $t[\texttt{sp}, G']$ is a minimal (multi)cut in $G'[\texttt{sp}]$. In both cases cases the test can be done in PTIME by simple reachability analysis in the graphs $G'[\texttt{sc}]$ and $G'[\texttt{sp}]$, 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 simple polytime reachability analysis in the graph as follows. To test whether S is a cut, we eliminate the edges in S from E, and then 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 that a set of edges is a minimal multicut.

Next, we study the problem of computing erase candidates $\text{ecand}(G, H)$ for the case where H has several triples. We need the following intermediate results.

Lemma 3. *Let G, H be ground RDF graphs in normal form. (i) If $E \in \text{ecand}(G, H)$, then there exists a triple $t_i \in H$ such that $E \in \text{ecand}(G, \{t_i\})$. (ii) If $D \in \text{dcand}(G, H)$, then there exists a triple $t_i \in H$ such that $D \in \text{dcand}(G, \{t_i\})$.*

Proof. (i) Suppose $G \not\models H$, then there is a triple $t_i \in H$ such that $G \not\models t_i$, which yields $\text{ecand}(G, H) = \{G\} = \text{ecand}(G, \{t_i\})$. Now we assume that $G \models H$. That is $H \subseteq \text{nf}(G)$. Let $T = (H \setminus I)$. T is non-empty because $I \not\models H$ and $\text{nf}(E) = E$. Now if T has more than one triple, then we add one triple of T to I and obtain $I' \in \text{ecand}(G, H)$ which is greater than I contradicting that E is maximal. Therefore T must have exactly one triple, say t_j. In this case can be easily verified that $E = \text{ecand}(G, \{t_j\})$. (ii) Follows directly from (ii).

The intuition of Lemma 3 is that each delete candidate in $\text{dcand}(G, H)$ is also a delete candidate of $\text{dcand}(G, \{t_i\})$ for some triple t_i in H. Therefore, the problem of computing delete candidates reduces to finding the minimal sets among the delete candidates associated to each triple in H. The following proposition formalizes this idea. For a set S of RDF graphs, we denote by $\texttt{Max}(S)$ and by $\texttt{Min}(S)$ the sets of graphs in S that are maximal and minimal, respectively.

Proposition 6. *Let G and H be ground RDF graphs. Then*
(i) $\text{ecand}(G, H) = \texttt{Max}(\bigcup_{t_i \in H} \text{ecand}(G, \{t_i\}))$.
(ii) $\text{dcand}(G, H) = \texttt{Min}(\bigcup_{t_i \in H} \text{dcand}(G, \{t_i\}))$.

Proof. (Sketch) Follows from Lemma 3(ii) and from the condition that $\text{ecand}(G, H)$ and $\text{dcand}(G, H)$ contains maximal and minimal graphs, respectively.

Proposition 6 yields a basic procedure for computing delete candidates. Find the minimal cuts for $\text{ecand}(G, \{t_i\})$ for each triple $t_i \in H$, and then find the minimum among them. The following theorem gives the complexity of the related decision problem.

Theorem 8. *Let G, H, E be ground RDF graphs. (The problem of deciding whether $E \in \text{ecand}(G, H)$ is in PTIME.*

Proof. (Sketch) Follows from Theorem 7 and Proposition 6. First, from Theorem 7 find in polytime whether E is an erase candidate in a set $\text{ecand}(G, \{t_i\})$ for some triple $t_i \in H$. If not $E \notin \text{ecand}(H)$. Otherwise, test whether E is minimal among $S = \bigcup_{t_i \in H} \text{ecand}(G, \{t_i\})$. In order to do so, for each triple $t' \in E$, test whether $E \setminus \{t'\}$ is not in the set S, which can be done by a set of $|H|$ polynomial tests, one for each triple in H.

5.3 General Graphs

We start this section by giving the complexity for the general case.

Theorem 9. *Let G, H, I be RDF graphs. (i)The problem of deciding whether $I \in \text{ecand}(G, H)$ is DP-complete. (ii) If G is in normal form the problem is coNP-hard and is in DP.*

Proof. (i) (DP-hard) Consider the following problem: given two RDF graphs without RDFS vocabulary G_1, G_2 test whether G_1 is the core of G_2. In [12] we proved that this problem is DP-complete. Consider the case where the graphs G, H do not mention RDFS vocabulary and also $G \not\models H$. In this case it is easily verified that I is an erase candidate of G, H iff I is the core of G. Indeed, in this case I should be $\text{nf}(G)$, which is equal to the core of G when G does not mention RDFS vocabulary. Therefore, testing whether I is an erase candidate of G, H is DP-hard. (DP) The test is equivalent to testing the following two conditions: (a) there is no mapping from H to I; and (b) I is maximal. Testing (a) is coNP, since the complement implies finding a mapping from H to I, which is NP. For testing (b), we have that I is maximal iff (c) for all triple $t \in G \setminus I$, there is a mapping from H to $I \cup \{t\}$. Testing (c) is in NP. Therefore the problem is also in DP.
(ii) (coNP-hard) If $E = G$ the problem is equivalent to testing that there are no mappings from H to G, which is coNP-complete [12]. Membership in DP can be proved using the argument of prove (i)(DP).

Given an RDF graph G, define G_* (the free graph of G) as the Herbrand model of G. And given a Herbrand model H of G, we denote by H^* the graph G.

Proposition 7. *Let G, H be RDF graphs. If $G = \text{nf}(G)$ we have $\text{ecand}(G, H) = (\text{ecand}(G_*, H))^*$.*

Proof. If $G = \text{nf}(G)$ we have $\text{nf}(G_*) = \text{nf}(G)_*$. Now, if E is an erase candidate for G, H, then E is the maximal subgraph of $\text{nf}(G)$ such that there is no map from H to E. Then E_* is the maximal subgraph of $\text{nf}(G)_* = \text{nf}(G_*)$ such that there is no map from H to E_*. Therefore E_* is an erase candidate of G_*, H. The converse can be verified by a similar argument, that is, if E_* is an erase candidate of G_*, H, E is an erase candidate of G, H.

Next, we reduce the problem of computing the erase candidates $\text{ecand}(G, t)$, where G is a ground graph and t a single triple, to computing erase candidates for ground graphs.

Proposition 8. *Let G be a ground RDF graph, t be an RDF triple (t may have blanks), and let $G' = \text{nf}(G)$, and let $A = \bigcup_{m_i : t \to G'} (\mu_i(t))$. Then:*
(i) $\text{dcand}_{\texttt{sc}}(G, H) = \texttt{MinCuts}(A[\texttt{sc}, G'], G'[\texttt{sc}])$.
(ii) $\text{dcand}_{\texttt{sp}}(G, H) = \texttt{MinCuts}(A[\texttt{sp}, G'], G'[\texttt{sp}])$

Proof. (Sketch) For simplicity, we consider that all delta candidates are of the form $D = \subseteq \text{dcand}_{\texttt{sc}}(G, t)$. The proof can be extended easily for the general case. We express Corollary 1 in terms of delta candidates is as follows. Let G, t, D be RDF graphs. Then $D \in \text{dcand}(G, t)$ iff D is a minimal cover set for $\texttt{minbases}(\text{nf}(G), t)$. The set $\texttt{minbases}(\text{nf}(G), t)$

contains all the triples associated to simple paths between pairs of nodes (u, v_i) in $A[\mathtt{sc}, G']$. So, the minimal covers for the collection $\mathtt{minbases}(\mathrm{nf}(G), t)$ are minimal multicuts in $\mathtt{MinCuts}(A[\mathtt{sc}, G'], G'[\mathtt{sc}])$, and vice versa.

Theorem 10. *Let G, E be RDF graphs. G is in normal form, and let H be a single triple. The problem of deciding whether $E \in \mathrm{ecand}(G, H)$ is in PTIME.*

PROOF. Follows from Theorem 8 and propositions 7 and 8. From Proposition 7 the problem is equivalent if we replace G by G_*, that is G is ground. The problem is also equivalent to testing whether $D = (G \setminus E)$ is in $\mathrm{dcand}(G, t)$. Let N_D be the set of edges associated to D in the graph $G[\mathtt{sc}]$, we define M_D similarly, but for the graph $G[\mathtt{sp}]$. From proposition 8, we have to prove that $N_D \in \mathtt{MinCuts}(A[\mathtt{sc}, G], G[\mathtt{sc}])$ and that $M_D \in \mathtt{MinCuts}(A[\mathtt{sp}, G], G[\mathtt{sp}])$. Each of the tests can be done in polytime, by reachability analysis on the respective graphs, in a similar manner to the test explained in the proof of Theorem 7. Just note that $A[\mathtt{sc}, G]$ cannot be greater than G, because t is a single triple pattern.

5.4 Further Approximations

We have so far studied the decision problem related to compute the set of erase candidates $\mathrm{ecand}(G, H)$. The set $\mathrm{ecand}(G, H)$ (respectively $\mathrm{dcand}(G, H)$ requires time $\Theta(2^{|G|})$ to be computed, even for the case where all the graphs are ground and H has a single triple. Indeed, the number of cuts is exponential, and so is the lower bound. Standard algorithms on cut enumeration for directed graphs can be adapted for our setting [21], but require exponential time.

We could further approximate the set of erase candidates by considering a more restricted notion of maximality, such a sets of maximal cardinality. In this case the problem of finding erase candidates, in some of the settings we have studied in this section, is equivalent to finding minimum k-edge cuts in directed graphs. A minimum k-edge cut is a multicut that separates each node v_i from u_i in each of k pairs of nodes (v_i, u_i). The problem has been thoroughly studied in [10]. For $k \leq 2$, the problem is in polytime by the applications of standard max-flow algorithms. For $k > 2$ the problem of deciding whether a given set is a minimum k-edge cut is co-NP complete.

Setting	Max-set	Max-card
G, H grounds, $\|H\| = 1$	PTIME	PTIME
G, H grounds	PTIME	PTIME
$G = \mathrm{nf}(G)$, $\|H\| = 1$	PTIME	coNP-complete
$G = \mathrm{nf}(G)$	coNP-hard, DP	coNP-hard, DP
General case	DP-complete	DP-complete

Table 3: Summary of the complexity of the problem of deciding whether $E \in \mathrm{ecand}(G, H)$.

Table 3 (first column) shows a summary of complexity results for the decision problem given in this section when the erase candidates are maximal under set inclusion (max-set). The last column shows the complexity of the decision problem if we approximate erase candidates using maximality under set cardinality (max-card). For the first two settings, the problem is equivalent to 1-edge cut and 2-edge cut (the 2-edge cut arises for the erase of triples of the form $(a, \mathtt{type}, b)$). In this cases, the complete set of erase candidates can be computed in polytime. This contrasts with the max-set case where, even though the decision problem is also in polytme, there is an exponential number of minimal cuts. For the third setting, from Proposition 8, it follows that the problem is equivalent to testing whether the associated delta candidate is k-edge minimal in each of the graphs $G[\mathtt{sc}]$ and $G[\mathtt{sp}]$, where in general $k > 2$. Therefore, for the max-card case, the problem is coNP-complete. The last two columns show the results of Theorem 9, which apply for the max-set and max-card cases.

6. RELATED WORK

Even though in this paper we have studied updates on RDF data, a complete review of related work must address updates on other data models closely related to RDF. For the sake of clarity, our discussion will be divided in three parts: (a) updates in knowledge bases and representation systems; (b) updates in graph databases; (c) updates in web databases: XML and RDF.

6.1 Updates in knowledge bases and representation systems

Relevant to our work is the problem of updating incomplete databases, that is, updating relational databases containing incomplete information and updates that are not completely specified. The semantics of an incomplete database 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 a triple $< S, rep, \Omega >$ where S is a set of tables, rep is a mapping from S to sets of complete instances, and Ω is a set of allowed operations (like insertion, join, and so on)

Abiteboul and Grahne [1], based on work of Imielinsky and Lipski [17], address strong and weak representation systems and generalize their notions for handling updates. 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. We are then in the presence of a weak representation system. The authors study three kinds of representation systems: Codd, naive, and the so-called C-tables (*conditional tables*). A Codd table is the usual relation with unknown (null) values. A naive table is a relation containing variables and constants. C-tables are tables with global and local conditions. A result by Imielinsky and Lipski [17] states that representation systems based on naive tables are *weak* if the set Ω includes standard relational operations but does not include negative selection nor set difference. In [1] this result is extended to consider updates. They show that for naive tables, when Ω is the same as above plus insertion, we have 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.

The relationship of the above theory with the problem addressed in this paper is that RDF can be considered an extension of a representation system based on the notion of naive tables without negation. Thus, in order to be appropriate for handling *update* and *erase*, RDF would need negation and disjunction.

6.2 Updates in graph databases

Updates have been also studied in the context of graph databases. This is relevant to our work because RDF represents graph-like data, and its model is closely related to graph data models [2]. Many proposals address querying graph databases [8, 14, 7]. Using labeled graphs as representation of a database schema an instance allows thinking database updates as transformations that can be expressed using pattern matching. This is the idea of the Graph-based data model (GDM) and its update language introduced by Hidders [16]. In GDM, an instance graph nodes represent object, composite values and values, while in a schema graph they represent classes. GDM supports inheritance through special edges labeled *isa*. Instance graphs where composite and value nodes are not unique are called *weak*. GUL, the update for GDM, is based on pattern matching. Two basic operations are defined: *addition* and *deletion*. *Addition* is specified using a *base* pattern that has to be matched, and an *extension* pattern indicating what nodes, edges and class names should be added when the base pattern is matched. For *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. These operations may result in weak instance graphs. The author solves this problem performing a *reduction*, an operation that merges nodes and yields an instance graph. The operations proposed in GUL have a syntactic flavor. From this point of view, the approach is a promising line to implement in RDF the semantic notions presented in this paper.

6.3 Updates in web databases: XML and RDF

XML Updates have been extensively addressed in the XML world. Tatarinov *et al* [28] proposed an XQuery extension that has been the first step leading to the proposal currently under study at the W3C [32]. In order to analyze the kinds of update operators needed in RDF, it is relevant to know what has been stated as necessary in XML. The W3C specified that update operators in XML *MUST,SHOULD and MAY* have some properties, like node deletion, insertion, and modification. For instance, the *MUST* properties for the XQuery Update Facility are: (a) be able to delete nodes; (b) be able to insert new nodes in specified positions; (c) be able to replace a node (d) be able to iterate over nodes to perform updates; (e) be able to compose update operators with other update operators; (f) be compositional with respect to XQuery expressions, that is, it may be possible to use an update wherever an XQuery expression is used.

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 [27] proposed five update operators, also based on [28]. These operators are: (a) *Add*: applies to nodes (subjects) as well as edges (predicates); (b) *InsertAfter*: applies to the rdf:Seq type container element only; (c) *Delete*: deletes all details about a resource (applies to nodes as well as edges); (d) *Remove*: applies only to a blank node that has a user-supplied blank node identifier, and slightly differs from the *Delete* operation;(e) *Replace*: replaces an old URI value, blank node identifier or a literal with a new URI value, blank node identifier or a literal respectively.

Zhan [33] proposed an extension to RQL. The operators considered are: (a) *Insert*, that includes edges (indicating properties, domains, and so on) or nodes. (b) *Delete*, which eliminates a triple from the existing model. This may imply removing an edge (property) or a node (if the deleted edge disconnects the node from the graph). (c) *Update* an existing triple. Updates are defined in an operational fashion, and semantic issues are considered to a very limited extent (e.g. if the inserted elements belong to a certain domain). Besides, the proposal does not account for blank nodes.

Another approach was proposed by Ognyanov and Kiryakov [24]. 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* [23] 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 language has three main features: (a) fine-grained granularity of the supported update primitives; (b) deterministic behavior of a sequence of update statements; (c) smooth integration with an underlying RDF query language. 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.

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 showed that our approach provides *syntax independence* in the sense of [11]. We characterized the update of a graph G with a graph H within the framework of the Katsuno-Mendelzon approach, and defined the meaning of the *update* and *erase* operations in RDF over a solid foundation. In the latter case, as we do not have negation nor disjunction in RDF, we provided an approximation to the Katsuno-Mendelzon postulates. Furthermore, we provided algorithms for calculating the update and erase operations and their approximations, including a detailed complexity analysis. Thus, we thoroughly studied the foundations of the semantics for updating RDF data.

Future work includes developing an update language for RDF, probably along the lines of transaction logic [5, 4], which seems a promising basis for the kinds of updates stud-

ied in this paper, given their transactional nature; extending our study to more expressive languages for the semantic web, like OWL; investigating other notions of distance (this notions, for example, may give give weight to triples based on their levels of trust).

8. REFERENCES

[1] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *Proceedings of the 11th International Conference on Very Lagre 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] G. Antoniou and F. van Harme. *A Semantic Web Primer*. MIT Press, London, England, 2004.

[4] A. Bonner and M. Kifer. An overview of transaction logic. *Theoretical Computer Science*, 1994.

[5] A. Bonner, M. Kifer, and M. Consens. Database programming in transaction logic. In *Proceedings of the Fourth International Workshop on Database Programming Languages (DBPL)*, pages 309–337, New York City, USA, 1993.

[6] D. Brickley and R.V.(Eds.) Guha. RDF vocabulary description language 1.0: RDF schema. *W3C Working Draft 23 January 2003*.

[7] M. P. Consens and A. O. Mendelzon. Graphlog: A visual formalism for real life recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 404–416, 1990.

[8] I. Cruz, A. Mendelzon, and P. Wood. A graphical query language supporting recursion. In *Proceedings of SIGMOD Conference*, pages 323–330, San Francisco, USA, 1987.

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

[10] N. Garg. Multicommodity flows and approximation algorithms. *Ph.D. Thesis. Department of Computer Science and Engineering, Indian Institute of Technology, Delhi*, 1994.

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

[12] C. Gutierrez, C. Hurtado, and A.O. Mendelzon. Foundations of semantic web databases. In *23rd. Symposium on Principles of Database Systems (PODS'04)*, pages 95–106, 2004.

[13] 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.

[14] M. Gyssens, J. Paredaens, and D. Van Gucht. A graph-oriented object database model. In *Proceedings of the Ninth ACM Symposium on Principles of Database Systems*, pages 417–424, 1990.

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

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

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

[18] H Katsuno and A. O. Mendelzon. On the difference between updating knowledge base and revising it. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 387–394, Cambridge, MA, 1991.

[19] 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.

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

[21] Hung-Yau Lin, Sy-Yen Kuo, and Fu-Min Yeh. Minimal cutset enumeration and network reliability evaluation by recursive merge and bdd. In *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003), 30 June - 3 July, Kiris-Kemer, Turkey*, 2003.

[22] 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 *Proceedings of the 12th. International Conference on World Wide Web*, pages 439–448, 2003.

[23] 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.

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

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

[26] R. Reiter. On specifying database updates. *Journal of Logic Programming 25(1)*, pages 53–91, 1995.

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

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

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

[30] World Wide Web Consortium. *Semantic Web*, 2001. http://www.w3.org/2001/sw/.

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

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

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