Coloring RDF Triples to Capture Provenance

Giorgos Flouris[1], Irini Fundulaki[1], Panagiotis Pediaditis[1,2], Yannis Theoharis[1,2], and Vassilis Christophides[1,2]

[1] Institute of Computer Science, FORTH, Greece
[2] Computer Science Department, University of Crete, Greece
{fgeo,fundul,pped,theohari,christop}@ics.forth.gr

Abstract. Recently, the W3C Linking Open Data effort has boosted the publication and inter-linkage of large amounts of RDF datasets on the Semantic Web. Various ontologies and knowledge bases with millions of RDF triples from Wikipedia and other sources, mostly in e-science, have been created and are publicly available. Recording provenance information of RDF triples aggregated from different heterogeneous sources is crucial in order to effectively support trust mechanisms, digital rights and privacy policies. Managing provenance becomes even more important when we consider not only explicitly stated but also implicit triples (through RDFS inference rules) in conjunction with declarative languages for querying and updating RDF graphs. In this paper we rely on *colored RDF triples* represented as quadruples to capture and manipulate explicit provenance information.

1 Introduction

Recently, the W3C Linking Open Data [29] effort has boosted the publication and interlinkage of large amounts of RDF datasets on the Semantic Web [1]. Various ontologies and knowledge bases with millions of RDF triples from Wikipedia [26] and other sources have been created and are available online [25]. In addition, numerous data sources in e-science are published nowadays as RDF graphs, most notably in the area of life sciences [27], to facilitate community annotation and interlinkage of both scientific and scholarly data of interest. Finally, Web 2.0 platforms are considering RDF and RDFS as non-proprietary exchange formats for the construction of information mashups [16,28].

In this context, it is of paramount importance to be able to store the *provenance of a piece of data* in order to effectively support trust mechanisms, digital rights and privacy policies. *Provenance* means *origin* or *source* and refers to *from where* and *how* the piece of data was obtained [33]. In the context of scientific communities, provenance information can be used in the proof of the correctness of results and in general determines their quality. In some cases, provenance of data is considered more important than the result itself.

The popularity of the RDF data model [8] and RDF Schema language (RDFS) [2] is due to the flexible and extensible representation of information, independently of the existence or absence of a schema, under the form of *triples*. An RDF triple, $(subject,property,object)$, asserts the fact that *subject* is associated with *object* through *property*. RDFS is used to add semantics to RDF triples, by imposing *inference rules* [15]

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 196–212, 2009.

(mainly related to the transitivity of subsumption relationships) which can be used to entail new *implicit* triples (i.e., facts) that are not explicitly asserted.

Currently, there is no adequate support for managing (i.e., querying and updating) provenance information of implicit RDF triples. There are two different ways in which such implicit knowledge can be viewed and this affects the semantics of update operations only. Under the *coherence* semantics [10], implicit knowledge does not depend on the explicit one but has a value on its own; therefore, there is no need for explicit "support" of some triple. Under this viewpoint, implicit triples are "first-class citizens", i.e., considered of equal value as explicit ones. On the other hand, under the *foundational* semantics [10] each implicit triple depends on the existence of the explicit triple(s) that imply it: implicit knowledge is only valid as long as the supporting explicit knowledge exists. It should be emphasized that the selected viewpoint is irrelevant as far as standard implication and querying is considered, but it affects the way updates should be performed; in particular, the coherence semantics corresponds to "belief set changes", whereas foundational semantics corresponds to "belief base changes", in the belief revision terminology [10].

In this paper we propose the use of *colors* (as in [4]) in order to capture the *provenance* of RDF *data* and *schema* triples. Intuitively, the color of an implicit and explicit triple represents the source from which the triple was obtained. We record the color of an RDF triple as a *fourth column*, hence obtaining an *RDF quadruple* as in [7,17].

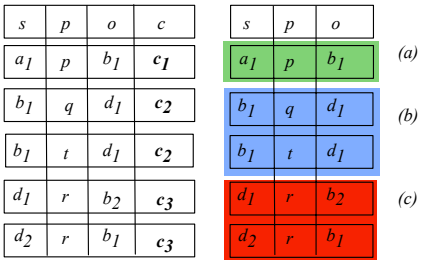


Fig. 1. Granularity Levels of Provenance

To provide the intuition of the granularity levels of provenance for RDF datasets that we capture with this work, we compare it with the representation of provenance information in the relational context. For instance, authors in [4] use colors to capture the provenance of relational tables, tuples and attributes. If we consider that a relational tuple of the form $[a_1{:}v_1, \ldots, a_k{:}v_k]$ with tuple identifier tid corresponds to a set of triples (tid, a_j, v_j), $j = 1, \ldots k$, then (see Fig. 1): a color assigned to *(a)* a single triple captures provenance at the level of *an attribute of the relational tuple*; *(b)* a collection of triples sharing the same subject captures provenance at the level of *the relational tuple* and finally *(c)* a set of triples whose subjects are instances of the same schema class, captures *provenance of the relational table*. The quadruples used to represent colored RDF/S triples leverage the syntax of RDF Named Graphs [5]: an RDF named graph can be modeled by arbitrary sets of triples sharing the same color.

The main contributions of this work are:

- We rely on the notion of *colors* to capture provenance information of explicit and implicit RDF triples. In particular, we employ a *semigroup* structure, defined by a set of colors to record and a binary operation "+" to reason over the provenance of RDF triples.

- We extend the RDFS inference rules [15] for computing the composite provenance of implicit RDF triples. In this respect, we devise an algorithm for determining *on the fly* the provenance of *non-materialized implicit* RDF triples.
- We study the semantics of *provenance propagation* and *querying* for the *subclass*, *subproperty* and *type* RDF hierarchies when colored RDF triples are represented as *quadruples*. In addition, we discuss atomic update operations (i.e., inserts and deletes) of RDF quadruples; updates are considered under the *coherence semantics* which allow us to preserve more knowledge during update operations [10].

The paper is structured as follows: in Section 2 we present the motivating example that we will use throughout this paper. Section 3 presents the basic RDF and RDFS notions. In Section 4 we introduce the notions of *color* and *quadruple* and discuss *inference* rules for sets of RDF quadruples. Section 5 discusses simple queries and atomic updates. We present related work in Section 6 and conclude in Section 7.

2 Motivating Example

We will use, for illustration purposes, an example taken from the News application domain. The RDFS schema of our News example captures information related to *newspapers* and *politicians* as well as the *relationships* between them and is contributed by different information sources.

	s	p	o	c
q_1	$\&NYT$	$endorses$	$\&B.\ Obama$	c_2
q_2	$\&NYT$	rdf:type	$Newspaper$	c_4
q_3	$Newspaper$	rdf:type	rdfs:Class	c_3
q_4	$Newspaper$	rdfs:subClassOf	$Mass\ Media$	c_3
q_5	$Mass\ Media$	rdfs:subClassOf	$Media$	c_5
q_6	$Candidate$	rdf:type	rdfs:Class	c_5
q_7	$\&B.\ Obama$	rdf:type	$Candidate$	c_5
q_8	$endorses$	rdf:type	rdf:Property	c_1
q_9	$endorses$	rdfs:domain	$Newspaper$	c_1
q_{10}	$endorses$	rdfs:range	$Candidate$	c_1
q_{11}	$Candidate$	rdfs:subClassOf	$Person$	c_1
q_{12}	$supports$	rdf:type	rdf:Property	c_1
q_{13}	$supports$	rdfs:domain	$MassMedia$	c_1
q_{14}	$supports$	rdfs:range	$Person$	c_1
q_{15}	$endorses$	rdfs:subPropertyOf	$supports$	c_2
q_{16}	$\&NYT$	$endorses$	$\&B.\ Obama$	c_1
q_{17}	$Media$	rdf:type	rdfs:Class	c_5

Fig. 2. Relation $\mathcal{Q}(s, p, o, c)$

Relation $\mathcal{Q}(s,p,o,c)$ that stores both schema and data triples is shown in Fig. 2. For $\mathcal{Q}(s, p, o, c)$, the s, p and o columns stand for the *subject*, *predicate* and *object* of an RDF triple. Column c is used to store the *provenance (color)* of a triple (s, p, o), which corresponds to the source from which this triple originates from and is represented by a URI (the URI of the source). We say that a triple $(s,\ p,\ o)$ is *colored* c iff $(s, p, o,\ c)$ is in $\mathcal{Q}(s, p, o, c)$. The set of triples (s, p, o) can be obtained by projecting on columns s, p and o of $\mathcal{Q}(s, p, o, c)$. In $\mathcal{Q}(s, p, o, c)$, a triple (s, p, o) can be assigned different colors (e.g., quadruples q_1 and q_{16}); this way, we can capture data integration scenarios in which the same piece of information originates from different sources.

Since RDF/S graphs can be seen as a kind of *node and edge labeled directed graphs*, we use the following graphical notation: *classes* and *properties* (binary relations between classes) of an RDFS vocabulary, are represented with boxes, and ovals respectively. *Instances* of classes contain their URI reference and to distinguish between

individual resources and classes, we prefix the URI of the former with the "&" symbol. RDFS built-in properties rdfs:subClassOf, rdf:type and rdfs:subPropertyOf are represented by dashed, dotted and dotted-dashed arrows respectively. Fig. 3 shows the graph obtained from the quadruples in $\mathcal{Q}(s, p, o, c)$ of Fig. 2. For (s, p, o, c), we color the edge with c and we draw $s \xrightarrow{p\ c} o$ (except if p is one of the built-in RDF/S properties where we draw $s \xrightarrow{c} o$).

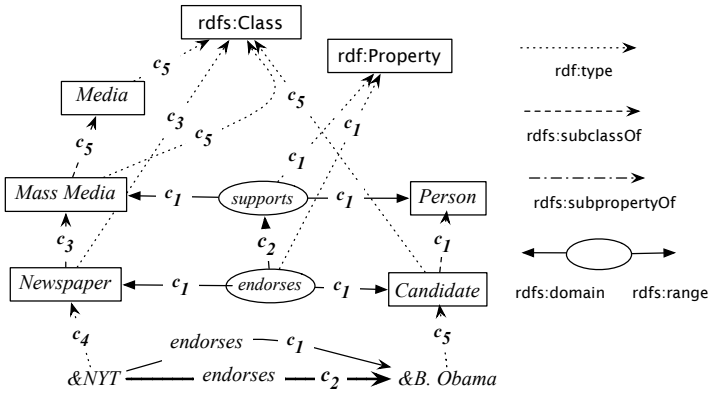


Fig. 3. Graph representation of $\mathcal{Q}(s, p, o, c)$

A great part of the information captured by a set of RDF triples can be inferred [15] by the *transitivity* of class and property subsumption relationships (rdfs:subClassOf and rdfs:subPropertyOf respectively) stated in the associated schemas. For instance, although not explicitly asserted in $\mathcal{Q}(s, p, o, c)$, we can infer that $Newspaper$ is a subclass of $Media$ because *(i)* $Newspaper$ is a subclass of $Mass\ Media$ (quadruple q_4) and *(ii)* $Mass\ Media$ is a subclass of $Media$ (quadruple q_5).

The question raised here is the following: *"what is the color of the implicit RDF triple?"*: the triple cannot be colored either c_3 or c_5 (colors of quadruples q_4 and q_5 respectively). In terms of provenance, we view the origin of this triple as *composite*: this triple should be colored c_3 *and* c_5. A possible way to represent this fact is to add quadruples ($Newspaper$, rdfs:subClassOf, $Media$, c_3) ***and*** ($Newspaper$, rdfs:subClassOf, - $Media$, c_5) in $\mathcal{Q}(s, p, o, c)$. But, in this case, the query *"return the triples colored* c_3*"* would falsely return ($Newspaper$, rdfs:subClassOf, $Media$). Hence, *composite origin cannot be captured by associating with the implicit triple separately the colors of its implying triples*.

Instead, we capture the provenance of implicit triples using colors defined by applying the special operation "+" on the colors of its implying triples. For instance, we color triple ($Newspaper$, rdfs:subClassOf, $Media$) with the color $c_{(3,5)} = c_3 + c_5$. Color $c_{(3,5)}$ can be seen as a *new, composite source of information*: it is a new URI and is assigned to the triples which are implied by triples colored c_3 and c_5 (as in the above example). Note that in this paper, we only consider *where* provenance [33], i.e., we record the sources that contributed in generating a particular triple, not the processes

(e.g., the particular inference rules) that led to its generation (*how* provenance) [33]. In Section 4 we discuss the properties of operation "+" in more detail.

As with annotated databases [11,12], we aim at supporting both *provenance propagation* and *provenance querying* over the subclass, subproperty and type RDF hierarchies. In the case of *provenance propagation*, queries are targeted to the original data and provenance is propagated to the query results. The result of these queries are *explicit* and *implicit* colored RDF triples. For instance, one can ask the query *"return all instances of class* $Newspaper$*"*. The result of the query will be ($\&NYT$, rdf:type, -$Newspaper$, c_4). Query *"return all subclasses of* $Media$*"* will return ($Newspaper$, rdfs:subClassOf, $Media$, $c_{(3,5)}$) and ($Mass\ Media$, rdfs:subClassOf, $Media$, c_5). Note that ($Newspaper$, rdfs:subClassOf, $Media$, $c_{(3,5)}$) is an *implicit* quadruple obtained by applying the transitive rdfs:subClassOf subsumption relationship on quadruples ($Mass\ Media$, rdfs:subClassOf, $Media$, c_5) and ($Newspaper$, rdfs:subClassOf, $Mass\ Media$, c_3) as previously discussed. On the other hand, in the case of *provenance querying*, queries are explicitly targeted at provenance information. Examples falling in this category are queries asking for the color of a given triple, or queries that filter triples given a color. For instance, the query *"return the color of (*$Newspaper$*,* rdfs:subClassOf*,* $Media$*)"*, will return $c_{(3,5)}$.

In this work, we support atomic updates; more specifically, one can either *(a)* insert a quadruple, or *(b)* delete an explicit or implicit quadruple. As previously stated, for our update operations we adhere to the coherence semantics [10], according to which we must explicitly retain all the implicit triples that will no longer be implied after the deletion of an explicit triple; we must also ensure that the resulting set of quadruples is redundant-free. As an example, consider the deletion of ($\&B.\ Obama$, rdf:type, $Candidate$, c_5). According to the coherence semantics we must retain the implicit information ($\&B.\ Obama$, rdf:type, $Person$, $c_{(1,5)}$) implied by quadruples ($\&B.\ Obama$, rdf:type, $Candidate$, c_5) and ($Candidate$, rdf:subClassOf, $Person$, c_1). Had we followed the foundational semantics, the implicit triple (and all the information it carries) would be lost, which would correspond to an unnecessary loss of information.

3 Preliminaries

In this work, we ignore non-universally identified resources, called *unnamed* or *blank nodes* [14]. In this respect, we consider two disjoint and infinite sets $\mathbf{U}$, $\mathbf{L}$, denoting the URIs and literals respectively.

Definition 1. *An RDF triple* $(subject,\ predicate,\ object)$ *is any element of the set* $\mathcal{T} = \mathbf{U} \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{L})$.

The RDF Schema (RDFS) language [2] provides a built-in vocabulary for asserting user-defined schemas in the RDF data model. For instance, RDFS names rdfs:Resource (res), rdfs:Class (class) and rdf:Property (prop)[1] could be used as objects of triples describing *class* and *property* types. Furthermore, one can assert *instance of* relationships of resources with the RDF predicate rdf:type (type), whereas *subsumption* relationships among classes and properties are expressed with the RDFS rdfs:subClassOf (sc)

[1] In parenthesis are the terms we use to refer to the RDFS built-in classes and properties.

and rdfs:subPropertyOf (sp) predicates respectively. In addition, RDFS rdfs:domain (domain) and rdfs:range (range) predicates allow one to specify the domain and range of the properties in an RDFS vocabulary. In the rest of this paper, we consider two disjoint and infinite sets of URIs of classes ($\mathbf{C} \subset \mathbf{U}$) and property types ($\mathbf{P} \subset \mathbf{U}$).

4 Provenance for RDF/S Data

In the same spirit as in [4] for relational databases, we assign a *color* to an RDF triple in order to capture its provenance.

Definition 2. *Structure (*$\mathbf{I}$*, "+") is a* commutative semigroup *where:*

- $\mathbf{I} \subset \mathbf{U}$ *is the set of colors, disjoint from the sets of class* $\mathbf{C}$ *and property types* $\mathbf{P}$.
- *"+" is a binary operation with the following properties:* $\forall c_1, c_2, c_3 \in \mathbf{I}$

$$\begin{array}{lll} c_1 + c_1 & = c_1 & (Idempotence) \\ c_1 + c_2 & = c_2 + c_1 & (Commutativity) \\ c_1 + (c_2 + c_3) & = (c_1 + c_2) + c_3 & (Associativity) \end{array}$$

Binary operation "+" is defined to capture the provenance of *implicit* RDF triples (see Table 1) and returns a *composite color* which is also in $\mathbf{I}$, i.e., if a triple t is implied by triple t_1 (whose color is c_1) and t_2 (whose color is c_2), then the color of t should be $c_1 + c_2$ and is denoted by $c_{(1,2)}$. In fact, $c_{(1,2)}$ is a new URI in $\mathbf{I}$ whose exact form is an implementation detail that we don't address in this paper. Note that, for n colors, there are $O(2^n)$ possible composite colors, so the storage of a composite color would require $O(n)$ bits in an efficient implementation. For an implicit RDF triple, its implying triples are those used to obtain it through the application of the inference rules that will be discussed in Section 4.1.

We say that color c_k is a *defining color* of $c_{(1,2,...,n)}$ and write $c_k \lessdot c_{1,2,...,n}$ iff $k \in \{1, 2, ..., n\}$, i.e., if $c_{(1,2,...,n)}$ can be obtained by an operation of the form $c_{(1,2,...,n)} = c_k + c$ for some color c. In other words, if a triple t is colored $c_{(1,2,...,n)}$ and c_k is a defining color of $c_{(1,2,...,n)}$, then t is an implicit triple which has been inferred using some triple colored c_k.

The intuition behind the properties of the "+" operation is the following: *(a)* an implicit RDF triple obtained from triples of the same color inherits the color of its implying triples (*idempotence*) *(b)* the color of an implicit triple is uniquely determined by the colors of the triples that imply it and not by the order of application of the inference rules (*commutativity* and *associativity*).

We should also note that $c_1 + c_2$ is a new color unless $c_1 = c_2$. In this manner, we keep a clear separation of what is given (explicit) and what can be implied (implicit). Moreover, the choice of idempotence is due to the semantics we give to colors as sources of information and the type of provenance we support: we need to know which sources participated in the creation of a new triple irrespective of which triples contributed to its implication. On the other hand, commutativity and associativity stem from the fact that the set of triples that are implied by a given triple set is the same irrespective of the order in which the inference rules are applied.

Definition 3. *An RDF quadruple* $(subject,\ predicate,\ object,\ color)$ *is any element of the set* $\mathcal{D} = \mathbf{U} \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{L}) \times \mathbf{I}$.

Using this definition, we can define the notion of an RDF dataset featuring triples associated with their provenance information as follows:

Definition 4. *An RDF Dataset* d *is a finite set of quadruples in* $\mathcal{D}$ *(*$d \subseteq \mathcal{D}$*).*

Note that none of the existing approaches in the literature combine intentional and extensional assignment of triples to provenance information (colors). In [30] RDF Named Graphs, capturing the provenance of RDF triples, are defined intentionally through SPARQL [31] views and do not support the explicit assignment of triples to such graphs, whereas in [5] a purely extensional definition is followed. The notion of colors as introduced in this paper allows us to capture both the intentional and extensional aspects of RDF graphs (i.e., sets of RDF triples) that are useful to record and reason about provenance information in the presence of updates.

4.1 Inference

Similarly to RDF graphs (i.e., sets of triples) we define a consequence operator that abstracts a set of inference rules. These rules compute the *closure* of an RDF dataset d, denoted by $Cn(d)$, which is obtained using the inference rules of Table 1. We say that a dataset d entails a quadruple q, and write $d \vdash q$, iff $q \in Cn(d)$. We say that a triple $t = (s,\ p,\ o)$ is colored c iff $(s, p, o, c)\ \in Cn(d)$.

The inference rules shown in Table 1 extend those specified in [15] in a straightforward manner to take into account colors. They compute the color of an implicit triple, using the colors of its implying triples. For instance, for our motivating example of Section 2, quadruple ($Newspaper$, rdfs:subClassOf, $Media$, $c_{(3,5)}$) is obtained by applying the *Transitivity of* sc $I_d^{(2)}$ rule on quadruples ($Newspaper$, rdfs:subClassOf, $Mass\ Media$, c_3) and ($Mass\ Media$, rdfs:subClassOf, $Media$, c_5).

4.2 Redundancy Elimination

In our work we consider that the RDF datasets are *redundant free*. An RDF dataset is redundant free if there does not exist a quadruple that can be implied by others when applying inference rules $I_d^{(1)}$ – $I_d^{(6)}$ of Table 1. The detection and removal of redundancies is straightforward using these rules.

In the sequel, we assume that queries and updates are performed upon redundant free RDF datasets. In effect, this means that redundancies are detected (and removed) at update time rather than at query time. This choice was made because we believe that in real scale Semantic Web systems, query performance should prevail over update performance. Redundant free RDF datasets were chosen because they offer a number of advantages in the case of transaction management for concurrent updates and queries.

Table 1. A subset of the RDFS Inference Rules for RDF Datasets

Reflexivity of sc

$$I_d^{(1)}: \frac{(C, \mathsf{type}, \mathsf{class}, c_1)}{(C, \mathsf{sc}, C, c_1)}$$

Transitivity of sc

$$I_d^{(2)}: \frac{(C_1, \mathsf{sc}, C_2, c_1), (C_2, \mathsf{sc}, C_3, c_2)}{(C_1, \mathsf{sc}, C_3, c_{(1,2)})}$$

Reflexivity of sp

$$I_d^{(3)}: \frac{(P, \mathsf{type}, \mathsf{prop}, c_1)}{(P, \mathsf{sp}, P, c_1)}$$

Transitivity of sp

$$I_d^{(4)}: \frac{(P_1, \mathsf{sp}, P_2, c_1), (P_2, \mathsf{sp}, P_3, c_2)}{(P_1, \mathsf{sp}, P_3, c_{(1,2)})}$$

Transitivity of class instantiation

$$I_d^{(5)}: \frac{(x, \mathsf{type}, C_1, c_1), (C_1, \mathsf{sc}, C_2, c_2)}{(x, \mathsf{type}, C_2, c_{(1,2)})}$$

Transitivity of property instantiation

$$I_d^{(6)}: \frac{(P_1, \mathsf{sp}, P_2, c_1), (x_1, P_1, x_2, c_2)}{(x_1, P_2, x_2, c_{(1,2)})}$$

5 Querying and Updating RDF Datasets

5.1 Querying RDF Datasets

In this section we discuss a simple class of queries that allow one to express queries on the *subclass*, *subproperty* and *type* hierarchies of an RDF dataset. We consider $\mathcal{V}$, $\mathcal{C}$ to be two sets of variables for resources and colors respectively; $\mathcal{V}$, $\mathcal{C}$, $\mathbf{U}$ (URIs) and $\mathbf{L}$ (literals) are mutually disjoint sets. We define a simple form of a quadruple pattern, called *q-pattern* which is an element from $(\mathbf{U} \cup \mathcal{V}) \times \{\mathsf{type} \cup \mathsf{sc} \cup \mathsf{sp}\} \times (\mathbf{U} \cup \mathcal{V}) \times (\mathbf{I} \cup \mathcal{C})$. In our context, a query is of the form (H, B, C) where H (head) is a q-pattern, B (body) is an expression defined after the antecedent of the inference rules presented in Section 4.1, and C (constraints) is a conjunction of *atomic predicates*. Each atomic predicate has the form: *(1)* $v \ = const$ for $v \in \mathcal{V}$; *(2)* $v \ \preccurlyeq \ v'$ for $v, v' \in \mathcal{C}$; *(3)* $c = c_1 + c_2 \ldots + c_k$ where $c \in \mathcal{C}$ and $c_i \in \mathbf{I}$.

According to the above definition, one can express constraints on resources *(1)*, on colors *(2)*, as well as to specify that a color considered in the query is defined by a set of other colors *(3)*. In addition, we require that all variables that appear in the head of the query (H) appear in the query's body (B). This restriction is imposed in order to have computationally desirable properties.

We denote variables with $?x$, $?y$, ... for resources and $?c_1$, $?c_2$, ... for colors. To define the query semantics, we use the notion of mapping (as in [21]) as follows: a *mapping* μ is a partial function $\mu : (\mathcal{V} \cup \mathcal{C}) \rightarrow \mathbf{U}$. The domain of μ ($dom(\mu)$) is the subset of $\mathcal{V} \cup \mathcal{C}$ where μ is defined. For a variable $?v$, $\mu(?v)$ denotes the resource or color to which $?v$ is mapped through μ.

To define the semantics of a *q-pattern* we must define first the *semantics of a property r over an RDF dataset d*, denoted by $[[r]]_d$. Given an RDF dataset d, $[[r]]_d$ for properties type, sc, sp, domain, range and user defined property p is given in Table 2. We write $d \vdash_S q$ to denote that dataset d entails quadruple q when the inference rules in S are applied on d.

Table 2. $[[r]]_d$ for type, sc, sp, domain, range and user defined property p

$$
\begin{array}{ll}
[[\mathsf{type}]]_d & = \{(x,\ y,\ c) \mid d \vdash_{\{I_d^{(2)}, I_d^{(5)}\}} (x, \mathsf{type}, y, c)\} \\
[[\mathsf{sc}]]_d & = \{(x,\ y,\ c) \mid d \vdash_{\{I_d^{(1)}, I_d^{(2)}\}} (x, \mathsf{sc}, y, c)\} \\
[[\mathsf{sp}]]_d & = \{(x,\ y,\ c) \mid d \vdash_{\{I_d^{(3)}, I_d^{(4)}\}} (x, \mathsf{sp}, y, c)\} \\
[[\mathsf{domain}]]_d & = \{(x,\ y,\ c) \mid d \vdash (x, \mathsf{domain}, y, c)\} \\
[[\mathsf{range}]]_d & = \{(x,\ y,\ c) \mid d \vdash (x, \mathsf{range}, y, c)\} \\
[[p]]_d & = \{(x,\ y,\ c) \mid d \vdash_{\{I_d^{(4)}, I_d^{(6)}\}} (x, p, y, c)\}
\end{array}
$$

Table 3. Semantics of *q-patterns*

$$
\begin{array}{ll}
[[(a, exp, ?y, ?c)]]_d & = \{\mu \mid dom(\mu) = \{?y, ?c\}\ and\ (a,\ \mu(?y),\ \mu(?c)) \in [[exp]]_d\} \\
[[(?x, exp, a, ?c)]]_d & = \{\mu \mid dom(\mu) = \{?x, ?c\}\ and\ (\mu(?x),\ a,\ \mu(?c)) \in [[exp]]_d\} \\
[[(?x, exp, ?y, c)]]_d & = \{\mu \mid dom(\mu) = \{?x, ?y\}\ and\ (\mu(?x),\ \mu(?y),\ c) \in [[exp]]_d\} \\
[[(a, exp, b, ?c)]]_d & = \{\mu \mid dom(\mu) = \{?c\}\ and\ (a,\ b,\ \mu(?c)) \in [[exp]]_d\} \\
[[(a, exp, b, c)]]_d & = \{\mu \mid dom(\mu) = \emptyset\ and\ (a,\ b,\ c) \in [[exp]]_d\}
\end{array}
$$

We write $\langle r \rangle_d$ to denote the semantics of property r when no inference rule is used. We can now define the semantics of a *q-pattern*. Consider an RDF dataset d and q = $(?X, exp, ?Y, ?c)$ a *q-pattern*, where exp is one of sc, sp, type, domain and range. Then the evaluation of q over d is defined as follows:

$$[[q]]_d = \{\mu \mid dom(\mu) = \{?X, ?Y, ?c\}\ and\ (\mu(?X),\ \mu(?Y),\ \mu(?c)) \in [[exp]]_d\}$$

In Table 3 we give the semantics of some *q-patterns* when URIs and colors are considered (where $a, b \in \mathbf{U}$ and $c \in \mathbf{I}$). Finally, given a mapping μ we say that μ satisfies an atomic predicate C, denoted by $\mu \vdash C$, per the following conditions:

$$
\begin{array}{ll}
\mu \vdash (?x = const) & \textit{iff}\ \mu(?x) = const, const \in \mathbf{U}, ?x \in dom(\mu) \\
\mu \vdash (?x = ?y) & \textit{iff}\ \mu(?x) = \mu(?y), ?x, ?y \in dom(\mu) \\
\mu \vdash (?c = ?c') & \textit{iff}\ \mu(?c) = \mu(?c'), ?c, ?c' \in dom(\mu) \\
\mu \vdash (?c \lessdot ?c') & \textit{iff}\ \mu(?c) \lessdot \mu(?c')\ ?c, ?c' \in dom(\mu) \\
\mu \vdash (?c = c_1 + ... + c_k) & \textit{iff}\ \mu(?c) = c_1 + \ldots + c_k, ?c \in dom(\mu)
\end{array}
$$

For a query $Q = (H, B, C)$ an RDF dataset d and mapping μ, such that $\mu \in [[B]]_d$, and $\mu \vdash C$, $\mu(H)$ is the quadruple obtained by replacing every variable $?x$ in $dom(\mu)$ with $\mu(?x)$. The color variable (if any) is replaced by the color obtained by applying the "+" operator as specified by the inference rules $I_d^{(1)}$ to $I_d^{(6)}$ of Table 1. The answer to Q is the union of the quadruples $\mu(H)$ for each such mapping μ.

5.2 Updating RDF Datasets

In this section we discuss *atomic* update operations (*inserts* and *deletes*). Recall that in our work we follow the *coherence semantics* [10] according to which we need to retain implicit information that would be lost during a triple deletion. Moreover, we enforce

that the resulting RDF datasets will remain valid with respect to the employed RDFS schema. The notion of validity has been described in various fragments of the RDFS language ([18,32]), and is used to overrule certain triple combinations. In the context of RDF datasets, the validity constraints are applied (and defined) at the level of the dataset, but the color-related part of the quadruple is not considered. The validity constraints that we consider in this work concern the *disjointness* between class, property and color names and the acyclicity of rdfs:subClassOf and rdfs:subPropertyOf subsumption relationships. An additional validity constraint that we consider in our work is that the subject and object of the instance of some property should be correctly classified under the domain and range of the property respectively. For a full list of the related validity constraints, see [19].

The semantics of each atomic update is specified by its corresponding *effects* and *side-effects*. The effect of an insert or delete operation consists of the straightforward insertion/deletion of the requested quadruples. The side-effects ensure that the resulting RDF dataset continues to be valid and non-redundant as discussed in [35]. Update semantics adhere to the principle of *minimal change* [9], per which a minimal number of insertions and deletions should be performed in order to restore a valid and non-redundant state of an RDF dataset. The effects and side-effects of insertions and deletions are determined by the kind of triple involved, i.e., whether it is a *class instance* or *property instance* insertion or deletion. Due to space restrictions we only describe *class instance* insertions and deletions.

INSERT **Operation.** A primitive insert operation is of the form: $\mathsf{insert}(s, p, o, i)$ where $s, p \in \mathbf{U}$, $o \in \mathbf{U} \cup \mathbf{L}$, $i \in \mathbf{I}$.

Algorithm 1. Class Instance Insertion Algorithm

Data: $\mathsf{insert}(x, \mathsf{type}, y, i)$, RDF dataset d
Result: Updated RDF dataset d
1 **if** $(\exists\ (x,\ y,\ i) \in [[\mathsf{type}]]_d)$ **then return** d;
2 **if** $(y \notin \mathbf{C})$ **then**
3 | **return** d;
4 **forall** $((x,\ z,\ i') \in \langle \mathsf{type} \rangle_d$ *s.t.* $\exists (y,\ z,\ i'') \in [[\mathsf{sc}]]_d$ *and* $i' = i + i'')$ **do**
5 | $d = d \setminus \{(x, \mathsf{type}, z, i')\}$;
6 **end**
7 $d = d \cup \{(x, \mathsf{type}, y, i)\}$;
8 **return** d

A formal description of the insertion of a quadruple (x, type, y, i) in an RDF dataset d along with its side-effects can be found in Algorithm 1. At line 1 we examine if the quadruple already belongs to the semantics of property type. If not, then we ensure that y is a class (lines 2–3). If it is, then we remove all class instantiation quadruples from the RDF dataset which can be implied through the quadruple to be inserted and the class subsumption relationships (lines 4–6), to guarantee that the result is redundant free. Finally, the quadruple is inserted (line 7). An example of a class instance insertion is shown in Figures 4(a) and 4(b).

Algorithm 2. Class Instance Deletion Algorithm

Data: $\mathsf{delete}(x, \mathsf{type}, y, i)$, RDF dataset d
Result: Updated RDF dataset d

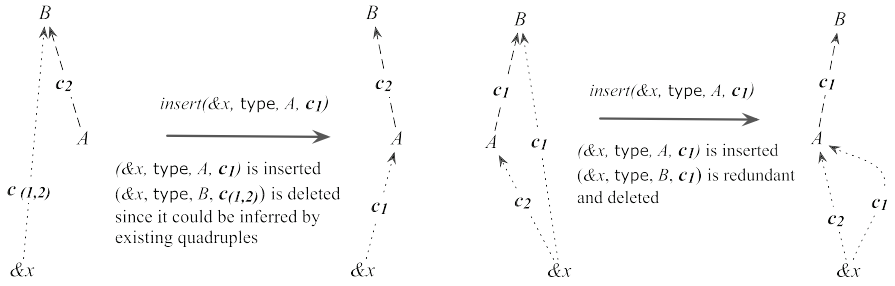
```
1  if (∄ (x, y, i) ∈ [[type]]_d) then return d;
2  forall ((x, y', i') ∈ [[type]]_d, (y', y, i'') ∈ [[sc]]_d s.t. i = i' + i'') do
3      forall (y', z, k) ∈ ⟨sc⟩_d s.t. y' ! = z do
4          if ∄(z, y, h) ∈ [[sc]]_d then d = d ∪ {(x, type, z, i' + k)}
5      end
6      d = d \ {(x, type, y', i')}
7  end
8  forall (x, o, h) ∈ ⟨q⟩_d, s.t. (q, c, i) ∈ ⟨domain⟩_d do
9      if ∄ (x, c, j) ∈ [[type]]_d then
10         d = d \ {(x, q, o, h)} ;
11         forall q' s.t. ∃ (q, q', h'') ∈ [[sp]]_d do
12             if ∃ (x, e, k) ∈ [[type]]_d s.t. ∃ (q, e, k') ∈ ⟨domain⟩_d then
               d = d ∪ {(x, q', o, h + h'')}
13         end
14
15 end
16 forall (o, x, h) ∈ ⟨q⟩_d, s.t. (q, c, i) ∈ ⟨range⟩_d do
17     if ∄ (x, c, j) ∈ [[type]]_d then
18         d = d \ {(o, q, x, h)} ;
19         forall q' s.t. ∃ (q, q', h'') ∈ [[sp]]_d do
20             if ∃ (x, e, k) ∈ [[type]]_d s.t. ∃ (q, e, k') ∈ ⟨range⟩_d then
               d = d ∪ {(o, q', x, h + h'')}
21         end
22
23 end
24 return d
```

DELETE **Operation.** A primitive delete operation is of the form: $\mathsf{delete}(s, p, o, i)$ where $s, p \in \mathbf{U}$, $o \in \mathbf{U} \cup \mathbf{L}$, $i \in \mathbf{I}$.

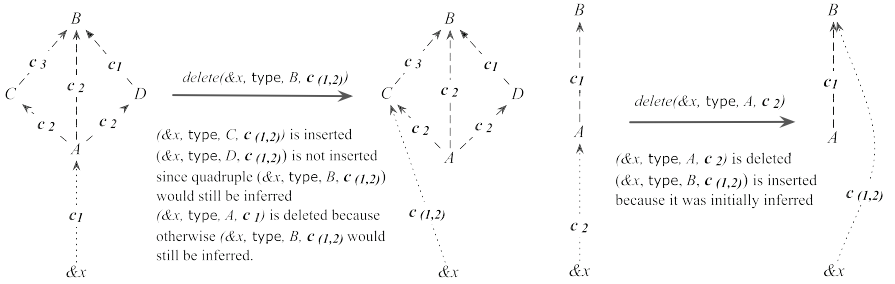
A formal description of the deletion of a quadruple (x, type, y, i) is given in Algorithm 2. At line 1 we examine if (x, type, y, i) belongs to the semantics of property type. If this is the case, then we must (1) insert all the quadruples that are implied by the quadruple that we wish to delete (per the coherence semantics) and (2) delete the quadruples that if retained would imply the quadruple we wish to delete (lines 2–7). To ensure that the RDF dataset is still valid after the updates, we must remove all properties originating from (or reaching resp.) x whose domain (or range resp.) is a class that x is no longer an instance of (lines 8–23). Examples of class instance deletions can be found in Figures 4(c) and 4(d).

5.3 Complexity Analysis

When working with colored RDF triples, one of the basic kinds of queries that we need to answer is *"what is the color of a triple"*. This is a *provenance query* and essentially boils down to finding, for a given triple $(s,\ p,\ o)$ and RDF dataset d, all quadruples of



(a) Class Instance Insertion (1) (b) Class Instance Insertion (2)



(c) Class Instance Deletion (1) (d) Class Instance Deletion (2)

Fig. 4. Examples of Class Instance Insertions ((a)–(b)) and Deletions ((c)–(d))

the form (s, p, o, c) in $Cn(d)$. Given that we work with redundant free RDF datasets (so implicit triples are not materialized), we present here an algorithm that computes the color of RDF triples *without* materializing $Cn(d)$, and discuss its complexity.

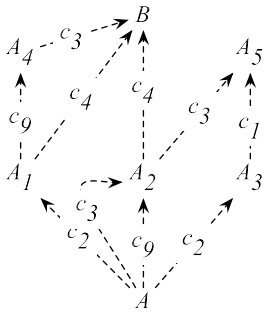


Fig. 5. sc Hierarchy

Consider an RDF dataset d whose size is N. In order to determine the color of $(s,\ p,\ o)$, without computing $Cn(d)$, we need to find all the possible ways in which a quadruple of the form (s, p, o, c) (for any c) can be inferred using quadruples from d. Using the algorithm below, this can be made in $O(N \log(N))$ time.

Certain quadruples are not involved in the inference rules in Table 1, so they cannot be implied by others: these are all the quadruples that *do not involve* the RDFS type, sc and sp relationships. Determining the color of such a quadruple is trivial, as we only need to check d (rather than $Cn(d)$), so it can be made in $O(\log(N))$ time by a simple search in d (using an appropriate index).

To determine the color of quadruples that involve the aforementioned built-in RDFS properties, we view the sc and sp hierarchies as *directed acyclic graphs*. Nodes in the graph are classes (properties resp.), and there exists an edge between nodes x and y iff x is a subclass (subproperty resp.) of y. The problem of obtaining the sequence of quadruples from which e.g., quadruple (A, sc, B, c) (for any c), can be implied is equivalent to discovering the path(s) (i.e., sequences of edges) in the DAG from node A to node B. For each of these sequences of edges, we keep the color of each involved quadruple. Recall that an edge may involve several quadruples (i.e., a triple can be colored with different colors). The algorithm uses a depth-first search and terminates when B is reached. At each step of the algorithm, we *(i)* find the superclasses of the context node (i.e., the node that we are currently looking at) that are also subclasses of the target node (e.g., B) and *(ii)* store the set of colors of the edges that we have already examined.

Algorithm 3. Traverse_Subsumption_Graph

```
   Data: Classes source and target, RDF Dataset d, Set of colors S;
   Result: Set of colors S;
 1 if source=target then
 2 |   return S;
 3 res ← ∅ ;
 4 foreach class x, where x is a superclass of source and subclass of target do
 5 |   Let (source, sc, x, c) ∈ d;
 6 |   Let S_x ← ∅ ;
 7 |   forall colors c_i in S do
 8 |   |   S_x ← S_x ∪ {c_i + c};
 9 |   end
10 |   res ← res ∪ Traverse_Subsumption_Graph(x,target,d,S_x);
11 end
12 return res
```

In Algorithm 3 in order to obtain all the classes that are superclasses of $source$ and subclasses of $target$ (line 4), we use the labeling scheme introduced in [6] that captures the subsumption relationships between classes and properties and allows us to determine whether a class (or property) is a subclass (or subproperty) of another in *constant time*. This is achieved by simply comparing the labels of the classes/properties. The labeling scheme is *solely used to prune irrelevant subclass and subproperty paths*, thereby limiting our search space significantly, without taking into consideration colors of triples.

As an example of application, consider the subclass hierarchy shown in Fig. 5. Suppose that we need to discover the color of triple $(A,\ \mathsf{sc},\ B)$. We start with class A and in the first step we will keep classes A_1,A_2 and sets of colors $S_{A_1} = \{c_2\}$, $S_{A_2} = \{c_3, c_9\}$ since the edges that determine that A is a subclass of A_1, A_2 are formed by triples colored $\{c_2\}$, $\{c_3, c_9\}$ respectively. For each of the superclasses of A (i.e., A_1, A_2), we perform exactly the same process and extend the sets of colors that we have obtained

using operation "+". Upon return of the recursion, these colors are placed into the set res to be returned. The output of the algorithm are the colors: $c_{(3,4)}$, $c_{(9,4)}$, $c_{(2,4)}$, $c_{(2,9,3)}$, where $c_{(1,\ldots,n)} = c_1 + \ldots + c_n$.

Given the fact that we prune subsumption paths that are not relevant in line 4 (using the labeling scheme), and that no cycles are allowed, the described process will, in the worst case, consider each quadruple in the RDF dataset d once. Given that each access requires a search in d, the cost of each access is $O(\log(N))$ (using an adequate indexing system). Therefore, the total complexity is $O(N \log(N))$.

For the other triples that may appear as implicit ones, the algorithm is similar. In particular, if t is of the form $(P,\ \mathsf{sp},\ Q)$, then the process is identical to the above, except that properties and subproperty relationships are considered instead of class and subclass relationships. If t is of the form $(A,\ \mathsf{type},\ B)$ then the process is almost identical, except that, in the first step of the recursion, we search for all classes whose explicit instance is A and are also (implicit or explicit) subclasses of B; the rest is the same. Finally, if t is of the form $(x,\ P,\ y)$, then, again, the process is identical, except that, in the first step of the recursion, we search for all explicit quadruples of the form (x, Q, y, c) such that Q is an explicit or implicit subproperty of P. The rest of the recursion steps are as in the above cases.

6 Related Work

So far, research on recording provenance for RDF data has focused on either associating triples with an *RDF named graph* [5,34] or by *extending* an RDF triple to a *quadruple* where the fourth element is a URI, a blank node or an identifier [7,17]. These works vary in the semantics of the fourth element which is used to represent *provenance*, *context* and *access control* information. *RDF Named graphs* have been proposed in [5,34] to capture *explicit provenance* information by allowing users to refer to specific parts of RDF/S graphs in order to decide *"how credible is"*, or *"how evolves"* a piece of information. An RDF named graph is a set of triples to which a URI has been assigned and can be referenced by other graphs as a normal resource; in this manner, one can assign explicit provenance information to this collection of triples.

Currently, there is no adequate support on how to manage provenance of *implicit* and explicit triples in the presence of *queries* and *updates*. Authors in [5] do not discuss RDFS inference, queries and updates in the presence of RDF named graphs. Unfortunately, existing declarative languages for querying and updating RDF triples have been extended either with RDF named graphs (such as SPARQL [23] and SPARQL Update [31]) or with RDFS inference support [22,24], but not with both. In this paper, we attempt to fill this gap by proposing a framework based on the use of colors to capture provenance of RDF triples and reason about the provenance of implicit triples for simple queries and atomic updates. In a previous work [20], we have introduced the notion of *RDF/S graphsets* which builds upon and extends the notion of RDF named graphs. In that paper we showed that the mechanism of RDF named graphs cannot capture the provenance of implicit RDF triples, and proposed RDF graphsets as a solution to this problem. In this paper, we use colors as an elegant and uniform way to capture the provenance of both explicit and implicit RDF triples. Colors are a generalization of

RDF named graphs [5]: a set of triples colored with a single color can be considered as belonging to the RDF named graph whose URI is the color (recall that colors are URIs). Colors obtained by applying the "+" operation on other colors simulate graph-sets [20]. The notion of colors as introduced in this paper allows us to capture both the intentional and extensional aspects of RDF graphs that are useful to record and reason about provenance information in the presence of updates whereas none of the existing approaches [5,30] combine intentional and extensional assignment of triples to provenance information.

On the other side of the spectrum, a significant amount of work on the issue has been done for relational and tree-structured databases [3,4,13,12]. Unlike these works, we consider both recursion and updates, whereas [13] does not consider updates, [3,4] supports updates but not recursion and [12] considers neither recursion or updates. Moreover, we do not focus on provenance propagation through relational operators but through inference rules that can be translated to relational unions and joins with bounded recursion. However, inference rules compute on the fly implicit triples without the need to materialize the provenance of intermediate results (per recursion step).

7 Conclusion

In this paper we proposed the use of colors to capture the provenance of RDF data and schema triples. We used the logical representation of quadruples to store the color of an RDF triple. The use of colors allows us to capture provenance at several granularity levels and can be considered as a generalization of RDF Named Graphs. One of the main contributions of the paper is the extension of RDFS inference rules to determine the provenance of *implicit* RDF triples, an extension which is not possible under the RDF named graphs approach and has been overlooked in the majority of approaches that deal with managing provenance information for RDF graphs. Note that the extended inference rules do not entail any additional complexity or scalability concerns to those already involving RDFS reasoning (see complexity analysis). As a future work we will study a more general algebraic structure (as in [13]) to capture the provenance of triples and mappings obtained by the SPARQL operators [23]. In addition, we plan to study the insertion and deletion of colors where the latter can be expressed as a batch deletion of quadruples sharing the same color.

Acknowledgements. This work was partially supported by the EU project CASPAR (FP6-2005-IST-033572).

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American-American Edition (2001)
2. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema (2004), `http://www.w3.org/TR/2004/REC-rdf-schema-20040210`
3. Buneman, P., Chapman, A.P., Cheney, J.: Provenance Management in Curated Databases. In: SIGMOD (2006)

4. Buneman, P., Cheney, J., Vansummeren, S.: On the Expressiveness of Implicit Provenance in Query and Update Languages. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 209–223. Springer, Heidelberg (2006)
5. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, Provenance and Trust. In: WWW (2005)
6. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the semantic web. In: WWW (2003)
7. Dumbill, E.: Tracking Provenance of RDF Data. Technical report, ISO/IEC (2003)
8. McBride, B., Manola, F., Miller, E.: B.M.: RDF Primer (February 2004), http://www.w3.org/TR/rdf-primer
9. Gardenfors, P.: Belief Revision: An Introduction. Belief Revision (29), 1–28 (1992)
10. Gardenfors, P.: The dynamics of belief systems: Foundations versus coherence theories. Revue Internationale de Philosophie 44, 24–46 (1992)
11. Geerts, F., den Bussche, J.V.: Relational Completeness of Query Languages for Annotated Databases. In: Arenas, M., Schwartzbach, M.I. (eds.) DBPL 2007. LNCS, vol. 4797, pp. 127–137. Springer, Heidelberg (2007)
12. Geerts, F., Kementsietsidis, A., Milano, D.: MONDRIAN: Annotating and Querying Databases through Colors and Blocks. In: ICDE. (2006)
13. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS (2007)
14. Gutierrez, C., Hurtado, C.A., Mendelzon, A.O.: Foundations of Semantic Web Databases. In: PODS (2004)
15. Hayes, P.: RDF Semantics (February 2004), http://www.w3.org/TR/rdf-mt
16. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid Prototyping of Semantic Mash-ups through Semantic Web Pipes. In: WWW (2009)
17. MacGregor, R., Ko, I.Y.: Representing Contextualized Data using Semantic Web Tools. In: Practical and Scalable Semantic Systems,conjunction with ISWC (2003)
18. Munoz, S., Perez, J., Gutierrez, C.: Minimal deductive systems for RDF. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 53–67. Springer, Heidelberg (2007)
19. Pediaditis, P.: Querying and Updating RDF/S Named Graphs. Master's thesis, Computer Science Department, University of Crete (2008)
20. Pediaditis, P., Flouris, G., Fundulaki, I., Christophides, V.: On Explicit Provenance Management in RDF/S Graphs. In: TAPP (2009)
21. Perez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
22. Perez, J., Arenas, M., Gutierrez, C.: nSPARQL: A Navigational Language for RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 66–81. Springer, Heidelberg (2008)
23. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (January 2008), http://www.w3.org/TR/rdf-sparql-query
24. PSPARQL, http://psparql.inrialpes.fr
25. DBLP Comp. Science Bibliography, http://www.informatik.uni-trier.de/~ley/db
26. DBPedia, http://www.dbpedia.org
27. Gene Ontology., http://www.geneontology.org
28. RDFizers, http://simile.mit.edu/wiki/RDFizers
29. W3C Linking Open Data, http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

30. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the Web. In: WWW (2008)
31. Seaborne, A., Manjunath, G.: SPARQL/Update: A language for updating RDF graphs (April 2008), http://jena.hpl.hp.com/~afs/SPARQL-Update.html
32. Serfiotis, G., Koffina, I., Christophides, V., Tannen, V.: Containment and Minimization of RDF/S Query Patterns. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 607–623. Springer, Heidelberg (2005)
33. Tan, W.C.: Provenance in databases: Past, current, and future. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering (2007)
34. Watkins, E., Nicole, D.: Named Graphs as a Mechanism for Reasoning About Provenance. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 943–948. Springer, Heidelberg (2006)
35. Zeginis, D., Tzitzikas, Y., Christophides, V.: On the Foundations of Computing Deltas Between RDF Models. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 637–651. Springer, Heidelberg (2007)