Annotation algebras for RDFS

Peter Buneman
University of Edinburgh
Email: opb@inf.ed.ac.uk

Egor V. Kostylev
University of Edinburgh
Email: ekostyle@inf.ed.ac.uk

***Abstract*—Provenance and annotation are intimately connected. On the one hand provenance can be regarded as a form of annotation, on the other hand, in order to understand how to convey annotations from source data to derived data, we need an account of *how* the data was derived – its provenance. Following a successful line of database research in which elements of a database are annotated with algebraic terms that describe the provenance of those elements, we develop an algebra of annotations for RDFS that differs from that developed for relational databases. We show how such an annotation algebra can be used for computing annotations on inferred triples that provide information on belief, trust and temporal aspects of data as well as providing a framework for default reasoning.**

I. Introduction

With the increasing interest in provenance in both databases and ontologies, there have been a number of proposals and systems for annotation of the underlying data with information concerning time, belief, and various aspects of provenance. The question that has been repeatedly posed in databases [17], [5], [2] is what happens to these annotations when a query is applied? Are they ignored or are they somehow passed through the query? In fact generic prototypes [4] and systems that are specific to some domain [6] have been developed for propagating annotations through queries. Suppose, for example, we take two tables S and T in which the individual tuples have been annotated. How should we annotate the tuples in the join of S and T? An obvious answer is to put on any output tuple the union of the annotations on the two contributing tuples. This does not always make sense; for example if the input tuples are annotated with the set of people who believe that this tuple is true or with the set of database versions for which the tuple is actually in the database, it might be more appropriate to annotate a join tuple with the intersection of the relevant annotations.

This problem has resulted in a variety of proposals for propagating annotations through queries. Notably, by using a semiring model for annotations, in [8] a tuple is annotated with a term in a semiring algebra that describes the provenance of the tuple – how it was formed by the operations of the relational query that constructed it. By suitable instatiatons operations of the semiring, one can realize various extensions to relational databases such as probabilistic databases, multiset semantics and certain kinds of constraint databases. The semiring model also generalizes a number of other models for provenance [5], [2].

Turning to ontologies, proposals have also been suggested for the annotation of RDF [13]. Named graphs [3] and temporal RDF [10] propose methods of adding annotations to RDF triples to express belief, trust, or temporal properties. Can we simply follow the work for relational databases in developing a general model for such annotations? Here we have to start by looking not at query languages for RDF, but at the inference rules for the ontologies such as those of RDFS [1]. Given annotations on the base triples, what should be the annotations on an inferred triple? Indeed in [15], [16], [12], with an initial goal applying fuzzy logic to RDFS proposes an algebra similar in many respects to that of [8]. In this paper we propose a somewhat more general – and possibly simpler – algebra to serve this purpose. Our proposals differ from the semirings in [8] in two ways. First, there are situations in which we do not want commutativity and second, while the number of triples inferred in an RDFS graph is always finite, the derivations can be unbounded. We therefore need an extra condition to prevent "infinite annotations". This condition precludes the possibility of bag semantics, which is useful for relational algbra but appears to be inapplicable to ontologies. Also in [8] there is a compact representation – a polynomial – of terms in the semiring algebra. In the algebra we develop, the compact representation is somewhat different.

To introduce annotation algebras, we give two simple examples of annotating an RDF graph shown in Fig. 1(a) (a dashed arrow represents a triple, which can be inferred from the rest of the graph).

Example 1: A temporal extension of RDF was introduced in [10], [9]. It consists of attaching to every RDF triple a set intervals that represents the times at which the triple is valid. An example of a temporal annotation of the graph is shown in Fig. 1(b): Picasso worked as a cubist from 1908 to 1919, paints are created by painters at least since engravings in Chauvet-Pont-d'Arc cave from about 29,000BC, and so on. The point here is that an annotation for the inferred triple was obtained by an intuitive calculation $(\{1908-1919\}\cap\{1906-1921\})\cup(\{1937\}\cap\{-29,000-Now\})=\{1908-1919,1937\}$.

Example 2: In [15] fuzzy annotations of RDF are used to describe degree of trust. To every triple a real in the range $[0,1]$ is attached. Such an annotation is shown in Fig. 1(c). The annotation for the inferred triple was calculated as $max(0.8*0.4, 0.3*1)=0.32$.

There is an obvious similarity between these examples: the calculations are both of the general form $(a\otimes b)\oplus(c\otimes d)$, and the calculations for the inferred triple are performed by suitably instantiating the operators $\oplus$ and $\otimes$. In fact [16], both of the annotation domains form so-called *BL-algebras*

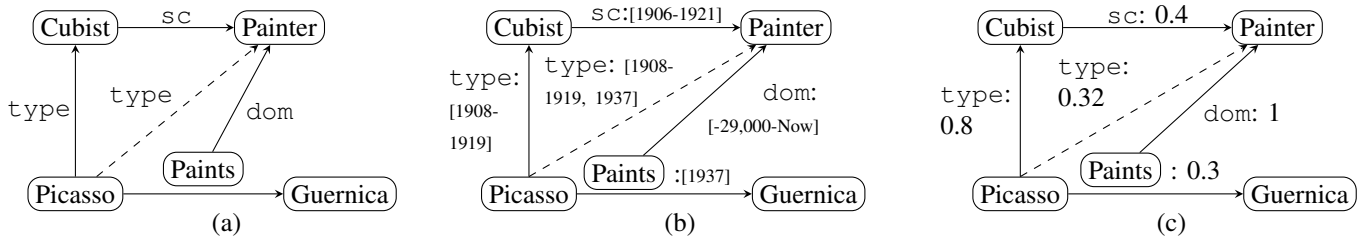

Fig. 1. Standard and annotated RDF graphs

[11], which in general preserve the properties of the deductive system [16].

Before describing these constructions in more detail we should briefly connect this work with the general issue of provenance. Our first observation is that an algebraic annotation or a triple of the form we have described is a synopsis of how that triple was derived – which is surely part of its provenance. The second is that exogenous provenance information such as who created a triple or when it was created can be added following the proposals of [3]. We would also like to compute such annotations for an inferred triple. Our proposals provide a method for tranferring the provenance annotations of explict triples to those that are derived.

Outline. This work has two aims. The first is to determine what algebraic structure is necessary for an annotation domain to keep the behavior of the deductive system the same as in the standard case. The second is to find "the most general" of such structures, which allows annotation of RDF graphs by elements of the general structure, apply inference rules, and then obtain annotations from specific domains on demand. In the following sections we review RDFS, introduce a new annotation algebra and provide some evidence that this is the appropriate algebra for RDFS. We give a freeness result that allows us to represent the terms of this annotation algebra and give some examples of the use of the algebra.

II. Preliminaries

Given a set of *RDF URI references* $\mathbf{U}$[1], let T be the set of *RDF triples* of the form $(s, p, o) \in \mathbf{U} \times \mathbf{U} \times \mathbf{U}$. Here s, p and o are called *subject*, *predicate* and *object* correspondingly. An *RDF graph* (or simply *graph*) is a finite set of triples $G \subseteq T$.

The RDF specification[13] includes RDF Schema (RDFS) [1] which is a vocabulary of reserved words designed to describe relationships between resources and properties. In this work we use the $\rho df = \{\mathtt{sp}, \mathtt{sc}, \mathtt{type}, \mathtt{dom}, \mathtt{range}\}$ fragment of RDFS [14]. The elements of ρdf represent sub-property, sub-class, domain, and range properties correspondingly. It is widely accepted that ρdf is a stable core of RDFS.

An *interpretation* of an RDF graph is a tuple $\mathcal{I} = (\Delta_R, \Delta_P, \Delta_C, P[\![\cdot]\!], C[\![\cdot]\!], \cdot^I)$ where

- Δ_R is a nonempty set of *resources*,
- Δ_P is a set of *property names* (not necessarily disjoint from Δ_R),
- $\Delta_C \subseteq \Delta_R$ is a set of *classes*,
- $P[\![\cdot]\!] : \Delta_P \to 2^{\Delta_R \times \Delta_R}$ is a *property extension* function,
- $C[\![\cdot]\!] : \Delta_C \to 2^{\Delta_R}$ is a *class extension* function,
- $\cdot^I : \mathbf{U} \to \Delta_R \cup \Delta_P$ is an *interpretation mapping*.

The interpretation $\mathcal{I}$ is a *model* for a graph G over ρdf, denoted by $\mathcal{I} \models G$, iff the conditions in Tab. I hold[2]. A graph G *entails* G', denoted $G \models G'$, if every model of G is a model of G'.

A sound and complete deductive system for entailment [14] is presented in Tab. II. An *instantiation* of an inference rule r from the system is a replacement of the variables $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}$, and $\mathcal{Y}$, occurring in the rule, by references from $\mathbf{U}$. If there is an instantiation $\frac{R}{R'}$ of r such that $R \subseteq G$, then the graph $G' = G \cup R'$ is the result of an *application* of r to G. A graph G' is *inferred* from G, denoted by $G \vdash G'$, iff G' is obtained from G by successively applying rules in Tab. II. This entailment can be checked by computing the *closure* of G, which is the maximal graph which can be inferred from G. It can be done in quadratic time [14].

III. Annotated RDFS

Definition 1: Given an algebra $\mathcal{K}$ with an elements set K containing a distinguished element $\bot$ a *$\mathcal{K}$-annotated RDF graph* (or simply an *a-graph*, if $\mathcal{K}$ is clear) is a function $G : T \to 2^K$ such that for each $t \in T$ holds $\bot \in G(t)$ and the set $Supp(G) = \{t : v \mid v \in G(t), v \neq \bot\}$ is finite.

If $v \in G(t)$ we write $t : v \in G$ and call it an *a-triple*. An a-graph G is *schema-acyclic*, if the subgraphs $\{(s, \mathtt{e}, o) : v \mid (s, \mathtt{e}, o) : v \in Supp(G)\}, \mathtt{e} = \mathtt{sc}, \mathtt{sp}$, do not contain non-trivial loops. The semantics for a-graphs is given in the following definitions.

Definition 2: A *$\mathcal{K}$-annotated interpretation* is a tuple $\mathcal{I} = (\Delta_R, \Delta_P, \Delta_C, P[\![\cdot]\!], C[\![\cdot]\!], \cdot^I)$ where $\Delta_R, \Delta_P, \Delta_C$ and $\cdot^I$ are the same as for the standard interpretation and $P[\![\cdot]\!], C[\![\cdot]\!]$ are defined as follows:

- for each $p \in \Delta_P$ holds $P[\![p]\!] : \Delta_R \times \Delta_R \to K$,
- for each $c \in \Delta_C$ holds $C[\![c]\!] : \Delta_R \to K$.

To define models for annotated RDFS we need more structure on the algebra $\mathcal{K}$. Let $\oplus$ and $\otimes$ be binary operations on K and $\bot$, $\top$ be distinct constants in it. For every $a, b \in K$

[1]For the sake of simplicity we do not consider blank nodes and literals. Their inclusion does not change the results of this work.

[2]The form of conditions in our definition of model is slightly different from that in [14], but they are equivalent *per se*. It is done to simplify the comparison with notion of model for annotated RDFS given in Sect. III.

(1) Simple interpretation:
- for each $(s,p,o) \in G$ holds $p^I \in \Delta_P$ and $(s^I, o^I) \in P[\![p^I]\!]$.

(2) Properties and classes:
- for each $\mathtt{e} \in \rho df$ holds $\mathtt{e}^I \in \Delta_P$;
- if $(x,y) \in P[\![\mathtt{sp}^I]\!]$ then $x, y \in \Delta_P$;
- if $(x,y) \in P[\![\mathtt{sc}^I]\!]$ then $x, y \in \Delta_C$;
- if $(x,y) \in P[\![\mathtt{type}^I]\!]$ then $y \in \Delta_C$;
- if $(x,y) \in P[\![\mathtt{dom}^I]\!]$ then $x \in \Delta_C$ and $y \in \Delta_P$;
- if $(x,y) \in P[\![\mathtt{range}^I]\!]$ then $x \in \Delta_C$ and $y \in \Delta_P$.

(3) Sub-property:
- $(p,p) \in P[\![\mathtt{sp}^I]\!]$;
- if $(p,q),(q,r) \in P[\![\mathtt{sp}^I]\!]$ then $(p,r) \in P[\![\mathtt{sp}^I]\!]$;
- if $(x,y) \in P[\![p]\!]$ and $(p,q) \in P[\![\mathtt{sp}^I]\!]$ then $(x,y) \in P[\![q]\!]$.

(4) Sub-class:
- $(c,c) \in P[\![\mathtt{sc}^I]\!]$;
- if $(c,d),(d,e) \in P[\![\mathtt{sc}^I]\!]$ then $(c,e) \in P[\![\mathtt{sc}^I]\!]$;
- if $x \in C[\![c]\!]$ and $(c,d) \in P[\![\mathtt{sc}^I]\!]$ then $x \in C[\![d]\!]$.

(5) Typing:
- $(x,c) \in P[\![\mathtt{type}^I]\!]$ iff $x \in C[\![c]\!]$;
- if $(x,y) \in P[\![p]\!]$ and $(c,p) \in P[\![\mathtt{dom}^I]\!]$ then $x \in C[\![c]\!]$;
- if $(x,y) \in P[\![p]\!]$ and $(c,p) \in P[\![\mathtt{range}^I]\!]$ then $y \in C[\![c]\!]$.

TABLE I
RDFS SEMANTICS

(1) Sub-property:
(a) $\dfrac{(\mathcal{A},\mathtt{sp},\mathcal{B})\ (\mathcal{B},\mathtt{sp},\mathcal{C})}{(\mathcal{A},\mathtt{sp},\mathcal{C})}$; (b) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y})\ (\mathcal{A},\mathtt{sp},\mathcal{B})}{(\mathcal{X},\mathcal{B},\mathcal{Y})}$.

(2) Sub-class:
(a) $\dfrac{(\mathcal{A},\mathtt{sc},\mathcal{B})\ (\mathcal{B},\mathtt{sc},\mathcal{C})}{(\mathcal{A},\mathtt{sc},\mathcal{C})}$; (b) $\dfrac{(\mathcal{X},\mathtt{type},\mathcal{A})\ (\mathcal{A},\mathtt{sc},\mathcal{B})}{(\mathcal{X},\mathtt{type},\mathcal{B})}$.

(3) Typing:
(a) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y})\ (\mathcal{A},\mathtt{dom},\mathcal{B})}{(\mathcal{X},\mathtt{type},\mathcal{B})}$; (b) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y})\ (\mathcal{A},\mathtt{range},\mathcal{B})}{(\mathcal{Y},\mathtt{type},\mathcal{B})}$.

(4) Sub-class Reflexivity:
(a) $\dfrac{(\mathcal{A},\mathtt{sc},\mathcal{B})}{(\mathcal{A},\mathtt{sc},\mathcal{A})\ (\mathcal{B},\mathtt{sc},\mathcal{B})}$; (b) $\dfrac{(\mathcal{X},\mathtt{e},\mathcal{A})}{(\mathcal{A},\mathtt{sc},\mathcal{A})}$ for $\mathtt{e} \in \{\mathtt{dom},\mathtt{range},\mathtt{type}\}$.

(5) Sub-property Reflexivity:
(a) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{A},\mathtt{sp},\mathcal{A})}$; (b) $\dfrac{}{(\mathtt{e},\mathtt{sp},\mathtt{e})}$ for $\mathtt{e} \in \rho df$; (c) $\dfrac{(\mathcal{A},\mathtt{sp},\mathcal{B})}{(\mathcal{A},\mathtt{sp},\mathcal{A})\ (\mathcal{B},\mathtt{sp},\mathcal{B})}$; (d) $\dfrac{(\mathcal{A},\mathtt{e},\mathcal{X})}{(\mathcal{A},\mathtt{sp},\mathcal{A})}$ for $\mathtt{e} \in \{\mathtt{dom},\mathtt{range}\}$.

TABLE II
RDFS DEDUCTIVE SYSTEM

we write $a \preceq b$ iff there exists $c \in K$ such that $a \oplus c = b$. The addition operation $\oplus$ will be used to combine annotations for the same triple and the $\bot$ constant represents the fact, that there is no information about a triple. The product operation $\otimes$ will be used to join annotations when applying inference rules and $\top$ represents the maximal annotation.

Definition 3: Let $\mathcal{K} = \langle K, \oplus, \otimes, \bot, \top\rangle$ be an algebra of type $(2,2,0,0)$. The $\mathcal{K}$-annotated interpretation $\mathcal{I}$ is a *model* for an a-graph G, denoted $\mathcal{I} \models G$, iff the conditions in Tab. III hold. An a-graph G *entails* H, denoted $G \models H$, if for every $\mathcal{I} \models G$ holds $\mathcal{I} \models H$.

By these definitions, each (non-annotated) RDF graph G can be considered as an a-graph $G' = \{t : \top \mid t \in G\} \cup E$, if $K = \{\bot, \top\}$ and $E = \{t : \bot \mid t \in T\}$. In this case, the definition of model for an a-graph coincides with the standard.

IV. A DEDUCTIVE SYSTEM FOR ANNOTATED RDFS AND DIOIDS

Definition 4: An algebra $\mathcal{K} = \langle K, \oplus, \otimes, \bot, \top\rangle$ is a *dioid* iff it is an idempotent semi-ring, i.e.

(1) $\langle K, \oplus, \bot\rangle$ is a semilattice, i.e. for each a, b, and c hold: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ (associativity), $a \oplus b = b \oplus a$, (commutativity), $a \oplus \bot = a$ (neutral element), $a \oplus a = a$ (idempotence);

(2) $\langle K, \otimes, \top\rangle$ is a monoid, i.e. for each a, b, and c hold: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$ (associativity), $a \otimes \top = a = \top \otimes a$ (neutral element);

(3) $\otimes$ is left and right distributive over $\oplus$, i.e. for each a, b, and c hold: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$, $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$;

(4) $\otimes$ is $\bot$-annihilating, i.e. for each a holds $\bot \otimes a = \bot = a \otimes \bot$.

A dioid is $\top$-*dioid* iff

(5) $\oplus$ is $\top$-*annihilating*, i.e. for each a holds $\top \oplus a = \top$.

Note, that $\top$-annihilation entails idempotence from (1).

An *instantiation* of a rule from the deductive system in Tab. IV is a replacement of variables $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}$, and Y by elements of $\mathbf{U}$, and variables v, v_1, v_2, and v_3 by elements of K, such that all relations for annotations hold. An *application* of a rule to an a-graph G and a *deduction* of an a-graph G' from G, denoted by $G \vdash G'$, is defined exactly the same way as for the standard case.

Note, that the system in Tab. IV differs from the one in Tab. II only by the presence of annotations and the generalisation rule $(*)$ which combines annotations for the same triple. Thus, that is natural to expect the new system to behave the same as the standard one. Particularly, they should coincide if $K = \{\bot, \top\}$. To obtain it we need some properties of $\mathcal{K}$.

Definition 5: Let $\mathcal{R}$ be the set of all inference rules of the form $\frac{\mathcal{T}_1\ \mathcal{T}_2}{\mathcal{T}}$ from Tab. IV and $Ins(r)$ the set of all instantiations of a rule $r \in \mathcal{R}$.

(1) A set of rules $\mathcal{R}' \subseteq \mathcal{R}$ is *associative*, iff for every $r, r' \in \mathcal{R}'$, $\frac{\tau_1\ \tau_2}{\tau_4}, \frac{\tau_1\ \tau_4'}{\tau_5'} \in Ins(r)$, and $\frac{\tau_4\ \tau_3}{\tau_5}, \frac{\tau_2\ \tau_3}{\tau_4'} \in Ins(r')$ holds $\tau_5 = \tau_5'$.

(2) A rule $r \in \mathcal{R}$ is *commutative*, iff for every $\frac{\tau_1\ \tau_2}{\tau} \in Ins(r)$ holds $\frac{\tau_2\ \tau_1}{\tau} \in Ins(r)$.

(3) A rule $r \in \mathcal{R}$ is *idempotent*, if $\frac{\tau\ \tau}{\tau} \in Ins(r)$ for every τ.

(4) A set of rules $\mathcal{R}' \subseteq \mathcal{R}$ is *left distributive* over $r \in \mathcal{R}$ if for every $r' \in \mathcal{R}'$, $\frac{\tau_2\ \tau_3}{\tau_4}, \frac{\tau_4'\ \tau_4''}{\tau_5'} \in Ins(r)$, and

(1) Simple interpretation:
- for each $(s,p,o) : v \in G$ holds $p^I \in \Delta_P$ and $v \preceq P[\![p^I]\!](s^I, o^I)$.

(2) Properties and classes:
- for each $\mathtt{e} \in \rho df$ holds $\mathtt{e}^I \in \Delta_P$;
- $P[\![\mathtt{sp}^I]\!](x,y)$ is defined only for $x,y \in \Delta_P$;
- $P[\![\mathtt{sc}^I]\!](x,y)$ is defined only for $x,y \in \Delta_C$;
- $P[\![\mathtt{type}^I]\!](x,y)$ is defined only for $y \in \Delta_C$;
- $P[\![\mathtt{dom}^I]\!](x,y)$ is defined only for $x \in \Delta_C$ and $y \in \Delta_P$;
- $P[\![\mathtt{range}^I]\!](x,y)$ is defined only for $x \in \Delta_C$ and $y \in \Delta_P$.

(3) Sub-property:
- $P[\![\mathtt{sp}^I]\!](p,p) = \top$,
- $P[\![\mathtt{sp}^I]\!](p,q) \otimes P[\![\mathtt{sp}^I]\!](q,r) \preceq P[\![\mathtt{sp}^I]\!](p,r)$;
- $P[\![p]\!](x,y) \otimes P[\![\mathtt{sp}^I]\!](p,q) \preceq P[\![q]\!](x,y)$.

(4) Sub-class:
- $P[\![\mathtt{sc}^I]\!](c,c) = \top$,
- $P[\![\mathtt{sc}^I]\!](c,d) \otimes P[\![\mathtt{sc}^I]\!](d,e) \preceq P[\![\mathtt{sc}^I]\!](c,e)$;
- $C[\![c]\!](x) \otimes P[\![\mathtt{sc}^I]\!](c,d) \preceq C[\![d]\!](x)$.

(5) Typing:
- $P[\![\mathtt{type}^I]\!](x,c) = C[\![c]\!](x)$;
- $P[\![p]\!](x,y) \otimes P[\![\mathtt{dom}^I]\!](c,p) \preceq C[\![c]\!](x)$;
- $P[\![p]\!](x,y) \otimes P[\![\mathtt{range}^I]\!](c,p) \preceq C[\![c]\!](y)$.

TABLE III
ANNOTATED RDFS SEMANTICS

(1) Sub-property:
(a) $\dfrac{(\mathcal{A},\mathtt{sp},\mathcal{B}) : v_1 \ (\mathcal{B},\mathtt{sp},\mathcal{C}) : v_2}{(\mathcal{A},\mathtt{sp},\mathcal{C}) : v_1 \otimes v_2}$;
(b) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v_1 \ (\mathcal{A},\mathtt{sp},\mathcal{B}) : v_2}{(\mathcal{X},\mathcal{B},\mathcal{Y}) : v_1 \otimes v_2}$.

(2) Sub-class:
(a) $\dfrac{(\mathcal{A},\mathtt{sc},\mathcal{B}) : v_1 \ (\mathcal{B},\mathtt{sc},\mathcal{C}) : v_2}{(\mathcal{A},\mathtt{sc},\mathcal{C}) : v_1 \otimes v_2}$;
(b) $\dfrac{(\mathcal{X},\mathtt{type},\mathcal{A}) : v_1 \ (\mathcal{A},\mathtt{sc},\mathcal{B}) : v_2}{(\mathcal{X},\mathtt{type},\mathcal{B}) : v_1 \otimes v_2}$.

(3) Typing:
(a) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v_1 \ (\mathcal{A},\mathtt{dom},\mathcal{B}) : v_2}{(\mathcal{X},\mathtt{type},\mathcal{B}) : v_1 \otimes v_2}$;
(b) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v_1 \ (\mathcal{A},\mathtt{range},\mathcal{B}) : v_2}{(\mathcal{Y},\mathtt{type},\mathcal{B}) : v_1 \otimes v_2}$.

(*) Generalisation:
$\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v_1 \ (\mathcal{X},\mathcal{A},\mathcal{Y}) : v_2}{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v_1 \oplus v_2}$.

(4) Sub-class Reflexivity:
(a) $\dfrac{(\mathcal{A},\mathtt{sc},\mathcal{B}) : v}{(\mathcal{A},\mathtt{sc},\mathcal{A}) : v \ (\mathcal{B},\mathtt{sc},\mathcal{B}) : v}$;
(b) $\dfrac{(\mathcal{X},\mathtt{e},\mathcal{A})}{(\mathcal{A},\mathtt{sc},\mathcal{A})}$ for $\mathtt{e} \in \{\mathtt{dom},\mathtt{range},\mathtt{type}\}$.

(5) Sub-property Reflexivity:
(a) $\dfrac{(\mathcal{X},\mathcal{A},\mathcal{Y}) : v}{(\mathcal{A},\mathtt{sp},\mathcal{A}) : v}$; (b) $\dfrac{}{(\mathtt{e},\mathtt{sp},\mathtt{e}) : \top}$ for $\mathtt{e} \in \rho df$; (c) $\dfrac{(\mathcal{A},\mathtt{sp},\mathcal{B}) : v}{(\mathcal{A},\mathtt{sp},\mathcal{A}) : v \ (\mathcal{B},\mathtt{sp},\mathcal{B}) : v}$; (d) $\dfrac{(\mathcal{A},\mathtt{e},\mathcal{X}) : v}{(\mathcal{A},\mathtt{sp},\mathcal{A}) : v}$ for $\mathtt{e} \in \{\mathtt{dom},\mathtt{range}\}$.

TABLE IV
ANNOTATED RDFS DEDUCTIVE SYSTEM

$\frac{\tau_1\ \tau_4}{\tau_5}, \frac{\tau_1\ \tau_2}{\tau_4'}, \frac{\tau_1\ \tau_3}{\tau_4''} \in Ins(r')$ holds $\tau_5 = \tau_5'$.

(5) A set of rules $\mathcal{R}' \subseteq \mathcal{R}$ is *right distributive* over $r \in \mathcal{R}$ if for every $r' \in \mathcal{R}'$, $\frac{\tau_1\ \tau_2}{\tau_4}, \frac{\tau_4'\ \tau_4''}{\tau_5'} \in Ins(r)$, and $\frac{\tau_4\ \tau_3}{\tau_5}, \frac{\tau_1\ \tau_3}{\tau_4'}, \frac{\tau_2\ \tau_3}{\tau_4''} \in Ins(r')$ holds $\tau_5 = \tau_5'$.

(6) A rule $r \in \mathcal{R}$ is *v-neutral*, $v \in K$, if for every $\frac{t_1:v\ t_2:v_2}{t_3:v_3} \in Ins(r)$ holds $v_2 = v_3$ and for every $\frac{t_1:v_1\ t_2:v}{t_3:v_3} \in Ins(r)$ holds $v_1 = v_3$.

(7) A rule $r \in \mathcal{R}$ is *v-annihilating*, $v \in K$, if for every $\frac{t_1:v\ t_2:v_2}{t_3:v_3} \in Ins(r)$ holds $v_3 = v$ and for every $\frac{t_1:v_1\ t_2:v}{t_3:v_3} \in Ins(r)$ holds $v_3 = v$.

Proposition 1: The set of inference rules $\mathcal{R}' = \{(1a),(1b),(2a),(2b),(3a),(3b)\}$ in Tab. IV is associative, the set of rules $\{(8)\}$ is associative, the rule (8) is commutative, idempotent and $\bot$-neutral, the set $\mathcal{R}'$ is left and right distributive over the rule (8), and each of the rules from $\mathcal{R}'$ are $\top$-neutral and $\bot$-annihilating iff $\mathcal{K}$ is a dioid.

Theorem 1 (Soundness and completeness): Given a dioid $\mathcal{K}$ and a-graphs G and H hold.

(1) If $G \vdash H$ then $G \models H$.
(2) If G is schema-acyclic and $G \models H$ then for every $t : v \in H$ there exists $v' \succeq v$ such that $G \vdash t : v'$.
(3) If $\mathcal{K}$ is $\top$-annihilating and $G \models H$ then for every $t : v \in H$ there exists $v' \succeq v$ such that $G \vdash t : v'$.

Hence, the deductive system behaves the same as the standard one iff $\mathcal{K}$ is a dioid for schema-acyclic a-graphs and a $\top$-dioid in general case.

As in the standard case, the important notion is the *closure* $cl(G) = \{\tau \mid G \vdash \{\tau\}\}$ of an a-graph G. To compute it we need a *representation* of an a-graph G, which is a finite set of a-triples R_G such that $R_G \cup E = G$, $E = \{t : \bot \mid t \in T\}$. The set $Supp(G)$ by the definition is a representation of G, but we also want to have a possibility to work with representations which contain an finite number of $\bot$-annotated triples.

Proposition 2: Let G be an a-graph and $\mathcal{K}$ a $\top$-dioid. For every representation R_G of G there exists an representation $R_{cl(G)}$ of $cl(G)$ which can be computed in polynomial time in the size of R_G if the complexities of $\oplus$ and $\otimes$ are polynomial bounded.

Let $\mathcal{K}$ and $\mathcal{K}'$ be two algebras. For any function $h : \mathcal{K} \to \mathcal{K}'$ and $\mathcal{K}$-annotated graph G denote $h(G)$ the set of $\mathcal{K}'$-annotated triples formed from G by applying h to each annotation.

Proposition 3: Let $h : \mathcal{K} \to \mathcal{K}'$ and $\mathcal{K}, \mathcal{K}'$ be $\top$-dioids. For every $\mathcal{K}$-annotated graph G the set $h(G)$ is a $\mathcal{K}'$-annotated graph and holds $cl(h(G)) = h(cl(G))$ iff h is a dioid homomorphism.

V. STRING DIOIDS FOR RDFS ANNOTATION

Prop. 3 enables us to obtain an a-graph from another one without recomputing annotations for inferred triples. The next step is to develop a "universal" annotation, i.e. an annotation from which we can obtain any other one by applying a corresponding dioid homomorphism.

Given an alphabet Σ define an *subsequence order* on the set of words Σ^*: $u \leq u'$ iff for some $u_1, \dots, u_n, w_1, \dots w_{n-1} \in \Sigma^*$ holds $u = u_1 u_2 \dots u_n$ and $u' = u_1 w_1 u_2 w_2 \dots w_{n-1} u_n$. A finite set $m \subseteq \Sigma^*$ is an *antichain* if for all $u \leq u'$, $u, u' \in m$, holds $u = u'$. Let $\min(m)$ be the set of minimal elements (w.r.t. $\leq$) of m. On the set of antichains $M[\Sigma]$ we define:

$$m_1 + m_2 = \min(m_1 \cup m_2),$$
$$m_1 \times m_2 = \min(\{w_1 w_2 \mid w_1 \in m_1, w_2 \in m_2\}),$$

Call $\mathcal{M}[\Sigma] = \langle M[\Sigma], +, \times, \emptyset, \{\epsilon\}\rangle$, where ϵ is the empty string, a *string dioid over generators* Σ. Note, that a string dioid is a $\top$-dioid. The following proposition says, that it is "the most general" of all $\top$-dioids.

Proposition 4: (1) Given a set of generators Σ the string dioid $\mathcal{M}[\Sigma]$ is the free $\top$-dioid on Σ, i.e. for any $\top$-dioid $\mathcal{K}_f = \langle K, \oplus, \otimes, \bot, \top\rangle$ and a valuation $\phi : \Sigma \to K$ there exists a unique homomorphism $\mathit{Eval}_\phi : M[\Sigma] \to K$ such that for each $a \in \Sigma$ holds $\mathit{Eval}_\phi(a) = \phi(a)$.
(2) The operations of the string dioid $+$ and $\times$ can be computed in polynomial time.

Hence, string dioids consititute an important and general subclass of $\top$-dioids. We now show how they can be applied to annotate RDF graphs. Let G be a $\mathcal{K}$-annotated graph and X a set of triple ids of triples from $Supp(G)$. We associate to G an "abstract" version which is a $(X \cup \emptyset)$-annotated graph $\bar{G}$ consisted of the same triples as G, but the annotation of each of them is its id if it has one, and $\emptyset$ otherwise.

Theorem 2: For every $\mathcal{K}$-annotated graph G holds $cl(G) = \mathit{Eval}_\phi \circ cl(\bar{G})$, where $\phi : X \to K$ is a valuation which associates the annotation of an a-triple in G to its id.

This theorem gives rise to the following strategy for annotating RDF graphs. Given a graph G we are to annotate it with elements from several domains. However, we would like to infer triples and their annotations only once, without recomputing the annotations for each annotation domain. In this case we construct an "abstract" version $\bar{G}$ of G by annotating triples with their ids. Then we can apply inference rules and obtain an abstract annotation from the string dioid over the set of ids for each triple. Finally, as soon as we need to get annotations from a specific domain we need to make sure that it is a $\top$-dioid, define a homomorphism by attaching specific annotations to triples in G and then apply it to abstract annotations of previously inferred triples.

VI. Applications to specific models

In this section we introduce several annotation domains for RDF graphs those are $\top$-dioids.

Temporal RDF [16]. This model treats the Ex. 1. The *temporal domain* $\mathcal{K}_T = \langle K_T, \oplus_T, \otimes_T, \bot_T, \top_T\rangle$ is defined as follows. Consider *temporal intervals* $[\alpha_1, \alpha_2]$ where $\alpha_{1,2} \in \mathbb{P} = \mathbb{Z} \cup \{-\infty, +\infty\}$, $\alpha_1 < \alpha_2$. Two intervals $[\alpha_1, \alpha_2]$ and $[\alpha_3, \alpha_4]$ are *adjacent* iff $\alpha_2 + 1 = \alpha_3$. The set K_T is the set of all pairwise disjoint and non-adjacent sets of intervals. On K_T a partial order is defined: $\gamma_1 \preceq \gamma_2$ iff for each $I_1 \in \gamma_1$ there exists $I_2 \in \gamma_2$ such that $I_1 \subseteq I_2$. For $\mathcal{K}_T$ we have: $\gamma_1 \oplus_T \gamma_2 = \inf\{\gamma \mid \gamma \succeq \gamma_i, i = 1, 2\}$, $\gamma_1 \otimes_T \gamma_2 = \sup\{\gamma \mid \gamma \preceq \gamma_i, i = 1, 2\}$, $\bot_T = \emptyset$ and $\top_T = \mathbb{P}$. The domain $\mathcal{K}_T$ forms an BL-algebra and hence a $\top$-dioid.

Fuzzy RDF [15] treats the Ex. 2. The domain here is $\mathcal{K}_F = \langle [0, 1], \max, \otimes_F, 0, 1\rangle$, where $\otimes_F$ is any t-norm of BL-algebra. If $\otimes_F$ is an ordinary multiplication as in Ex. 2, then the domain becomes probabilistic. As $\mathcal{K}_T$ domain, $\mathcal{K}_F$ is a $\top$-dioid.

Default RDF. In both of the previous domains the product operation in the dioid is commutative. Next we introduce an example of $\otimes$-noncommutative annotations.

Suppose we want to represent – in an RDF graph – information about an attribute attached to resources modelled by the graph. The straightforward way is to introduce a new property to the vocabulary of the graph and handle it as usual. Nevertheless, in some situations this way can be not optimal: If we have a broad system of classes and values of the attribute for subclasses and elements of a class are usually the same, then it is natural to keep the default attribute value for the class and the value for a resource only if it differs from usual one of the class it belongs to.

For a substantiation, consider an RDF graph denoted in Fig. 2 representing a (part of) botanical taxonomy[3]. The triples of this graph have annotations which store values of an attribute **PublishedBy** for every taxon in the graph. The division *Pinophyta* was introduced by Carl Linnaeus, so the triple $(Plantae, \mathtt{sc}, Pinophyta)$ is annotated by $Linnaeus$. The subsequent taxons down to family *Araucariaceae* keep this attribute value, so we annotate all sc-triples between *Pinophyta* and *Araucariaceae* with a special value $\star$ which means "derived from above". The same value keeps for genus *Araucaria* and species *Araucaria Araucana*. But genus *Wollemia* was introduced by David Noble as well as its only species *Wollemia Nobilis*, so the annotations for the triples $(Araucariaceae, \mathtt{sc}, Wollemia)$ and $(Wollemia, \mathtt{sc}, W.Nobilis)$ are $Noble$ and $\star$ correspondingly. Thus, to find out from the graph who introduced a taxon we need to go up from its node along the tree until we find an edge with an annotation differs from $\star$. An annotation value for a triple here represents not the value of the attribute by itself, but its *difference* with the value of higher triple in the tree. Hence, it is natural to phisically keep such annotations in a memory only it they differ from $\star$. That is why if the value of an attribute for a resource in most cases is determined by a class it belongs to and the number of attributes are large, the advantage in memory can be sufficient.

Before we define the default domain formally, note, that the situation in general case can be more complicated than in the latter example, because a ρdf subgraph of a graph does not nesessarily have a tree structure. Therefore, in certain cases we need to combine different annotations for a triple. To this effect we require a set of attirbute values to have a union operation and the lowest element, i.e. to be a semilattice. In many cases (as in the taxonomy example) this set has no structure, but it is possible to enrich it with the lowest element `Unknown` and the greatest element `Error`. The latter is justified by situations

[3]This is just an example and the actual information may be incorrect.

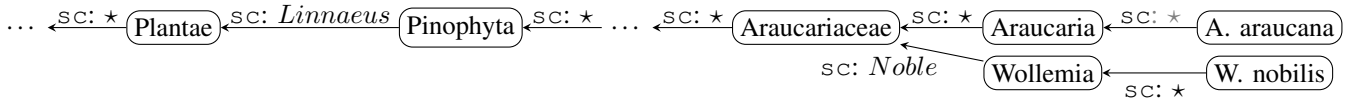

Fig. 2. The RDF graph representing botanical taxonomy

when from one source we get an attribute value for a resource, but from another source we get a different value for the same resource which contradicts with the first one. The union of this information is inconsistent and requires a manual intervention, which can be flagged by `Error` annotation.

Let $\mathbf{A}$ be an attribute with a domain $\langle A, \sqcup, 0\rangle$ which is a semilattice with union $\sqcup$ and the lowest element 0. Consider an $\mathbf{A}$-*default* domain $\mathcal{K}_\mathbf{A} = \langle A^\star \cup \{\bot^\star, \top^\star\}, \oplus_\mathbf{A}, \otimes_\mathbf{A}, \bot^\star, \top^\star\rangle$, where $A^\star = (A \times \{True, False\})$ and $\bot^\star$, $\top^\star$ are new special symbols. Note, that the meanings of $\bot^\star$ and an annotation with 0 are different: The first one says that a triple is not in the support of a graph and the second says that the value of the attribute is minimal according to $\sqcup$. The second boolean component of $A^\star$ corresponds to $\star$ in the taxonomy example above, i.e. it is $True$ in an annotation for a triple iff the actual value of the attribute is the union of the first component of the annotation with values those can be derived from triples above, and $False$ overwise. The operations are defined as follows:

$$(a_1, b_1) \oplus_\mathbf{A} (a_2, b_2) = (a_1 \sqcup a_2, b_1 \vee b_2),$$
$$(a_1, b_1) \otimes_\mathbf{A} (a_2, b_2) = \begin{cases} (a_1 \sqcup a_2, b_2) \text{ iff } b_1 = True, \\ (a_1, b_1) \text{ iff } b_1 = False. \end{cases}$$

These operations extend to elements $\bot^\star$ and $\top^\star$ in a way to keep the properties of $\bot$ and $\top$ in the dioid. Next, we describe the meaning of this operations. The addition $\oplus_\mathbf{A}$ is applied when we union annotations about the same triple. Hence, the values a_1 and a_2 are joined by the semilattice operation $\sqcup$ and the possibility to derive values from a ρdf structure above exists only if it exists in any of the considering annotations. The product $\otimes_\mathbf{A}$ is applied when we infer triples by transitivity of `sc` or similar inference rule from Tab. IV. If $b_1 = True$ we union the current value a_1 with the derived value a_2 and the possibility of father deriving depends on b_2. If $b_1 = False$, we just keep the annotation (a_1, b_1) which override any annotation from a structure above. Finally, $\mathcal{K}_\mathbf{A}$ can be easily checked to be $\otimes$-noncommutative $\top$-dioid.

To use the $\mathbf{A}$-default dioid $\mathcal{K}_\mathbf{A}$ for storing values of an attribute $\mathbf{A}$ for resources we assume $(0, True)$-annotation for a triple by default and do not keep it in a memory. As soon as we need the real value of the attribute for a resource a we infer a triple $(a, \texttt{type}, r)$ and get the first component of annotation for it. (Here r is the *root* of ρdf subgraph of the considered graph; if it does not exist, we can always introduce it.)

VII. Conclusions

Although annotation algebras have been studied for database query languages [8] and have also recently been investigated for RDF query languages [7], we have suggested an alternative and more natural algebra for the annotation of RDFS, and we have given some examples of its use. There may be some mileage to be gained by combining these two algebras. One of the applications of the proposed algebra is a system for default reasoning about certain annotations on RDF resources, which may also prove to be a useful mechanism for physically storing those annotations. We would like to find similar mechanisms for efficiently storing default annotations on triples.

We are grateful to Marcelo Arenas, Boris Motik, Floris Geerts and Alex Simpson for helpful discussions. This work was supported by a grant from the UK Engineering and Physical Sciences Research Council.

References

[1] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. http://www.w3.org/TR/rdf-schema/, Feb. 2004.

[2] P. Buneman, S. Khanna, and W. Tan. Why and Where: A Characterization of Data Provenance. In *ICDT 2001*, volume 1973 of *LNCS*, pages 316–330. Springer, 2001.

[3] J. J. Carroll, C. Bizer, P. J. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW*, pages 613–622, 2005.

[4] L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. DBNotes: a post-it system for relational databases based on provenance. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 942–944, New York, NY, USA, 2005. ACM.

[5] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, 2000.

[6] R. D. Dowell, R. M. Jokerst, A. Day, S. R. Eddy, and L. Stein. The Distributed Annotation System. *BMC Bioinformatics*, 2:7, 2001.

[7] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring RDF Triples to Capture Provenance. In *International Semantic Web Conference*, pages 196–212, 2009.

[8] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, New York, NY, USA, 2007. ACM.

[9] C. Gutierrez, C. A. Hurtado, and A. Vaisman. Introducing Time into RDF. *IEEE Trans. on Knowl. and Data Eng.*, 19(2):207–218, 2007.

[10] C. Gutiérrez, C. A. Hurtado, and A. A. Vaisman. Temporal RDF. In *ESWC*, pages 93–107, 2005.

[11] P. Hájek. *Metamathematics of Fuzzy Logic (Trends in Logic)*. Springer, 1 edition, November 2001.

[12] A. Hogan, G. Lukacsy, N. Lopes, A. Polleres, U. Straccia, A. Zimmermann, and S. Decker. RDF Needs Annotations. In *Proceedings of W3C Workshop — RDF Next Steps*. W3C, 2010.

[13] F. Manola, E. Miller, and B. McBride. RDF Primer, W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax/, Feb. 2004.

[14] S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal Deductive Systems for RDF. In *ESWC*, pages 53–67, 2007.

[15] U. Straccia. A Minimal Deductive System for General Fuzzy RDF. In *Proceedings of the 3rd International Conference on Web Reasoning and Rule Systems (RR-09)*, number 5837 in Lecture Notes in Computer Science, pages 166–181. Springer-Verlag, 2009.

[16] U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres. A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*. AAAI Press, 2010.

[17] Y. R. Wang and S. E. Madnick. A Polygen Model for Heterogeneous Database Systems: The Source Tagging Perspective. In *VLDB*, pages 519–538, 1990.