A General Framework for Representing and Reasoning with Annotated Semantic Web Data

Umberto Straccia
ISTI - CNR
Pisa, Italy

Nuno Lopes, Gergely Lukácsy, and Axel Polleres
Digital Enterprise Research Institute
National University of Ireland, Galway

Abstract

We describe a generic framework for representing and reasoning with annotated Semantic Web data, formalise the annotated language, the corresponding deductive system, and address the query answering problem. We extend previous contributions on RDF annotations by providing a unified reasoning formalism and allowing the seamless combination of different annotation domains. We demonstrate the feasibility of our method by instantiating it on (i) temporal RDF; (ii) fuzzy RDF; (iii) and their combination. A prototype shows that implementing and combining new domains is easy and that RDF stores can easily be extended to our framework.

Introduction

RDF[1] is a Semantic Web representation language in which the basic ingredients are triples of the form (s, p, o), stating that a subject s has a property p with value o. The need of allowing triples to be annotated with a term, such as time (Gutierrez, Hurtado, Vaisman 2007; Pugliese, Udrea, Subrahmanian 2008; Tappolet, Bernstein 2009), degree of truth (Straccia 2009), trust (Hartig 2009), and provenance (Dividino et al. 2009) is emerging. These extensions demand a need to deal with quadruples rather than with triples, where the additional term has some specific semantics and operational behaviour. Along these lines, this paper contributes as follows: (i) we describe a general framework for annotated RDF and RDF Schema (RDFS), in which annotations can be taken from a family of algebraic structures. We allow RDF triples τ to have annotation terms v, resulting in annotated triples of the form $\tau : v$, where v belongs to a set of elements L on which some specific operations are defined. *E.g.*, in the temporal case L may be a set of time intervals, while in the fuzzy case L is $[0, 1]$; (ii) our work extends previous contributions on RDF annotations by providing both a unified reasoning formalism and the seamless combination of different annotation domains; (iii) we show that RDF stores can easily be extended to our framework without an increase in reasoning complexity (provided that the operations on the domain are polynomial); (iv) we demonstrate the feasibility of our method by giving a formal description on how we can represent and reason within RDFS with temporal and fuzzy annotations, and their combination; and (v) we discuss the integration of annotated triples with standard, non-annotated triples and SPARQL.

[1] http://www.w3.org/RDF/.

Related work: Closest to our work is (Straccia 2009), that describes fuzzy RDF in a general setting where triples are annotated with a degree of truth in $[0, 1]$. For instance, "Rome is a big city to degree 0.8" can be represented with $\langle Rome, \mathsf{type}, BigCity\rangle : 0.8$; the annotation domain is $[0, 1]$. We further generalise this work, by allowing more general domains, while following a similar approach to the formalisation of the semantics and description of inference rules. Another related work (Udrea, Recupero, Subrahmanian 2006) allows to annotate triples with truth values taken from a *finite partial order*. Here, triples are of the form $(s, p : v, o)$, where the property, rather than the triple is annotated. We instead rely on a richer, not necessarily finite, structure and provide additional inference capabilities to (Udrea, Recupero, Subrahmanian 2006), such as a more involved propagation of annotation terms through schema triples. For instance, in the temporal domain, from $\langle a, \mathsf{sc}, b\rangle : [2, 6]$ and $\langle b, \mathsf{sc}, c\rangle : [3, 8]$, we will infer $\langle a, \mathsf{sc}, c\rangle : [3, 6]$ (sc is the subclass property). Essentially, Udrea et al. do not provide an operation to combine the annotation in such inferences, while the algebraic structures we consider support such operations instead. Also, it requires specific algorithms, while we show that a simple extension to the classical RDF inference rules suffices.

Next, we (i) recap basic notions of classical RDF(S) we rely on; (ii) formalise our annotated language, the deductive system and address the query answering problem; (iii) show the feasibility of our approach for two specific cases, the fuzzy and the temporal case; (iv) provide a description on how to combine multiple domains, on possible options to handle non-annotated triples and SPARQL; and (v) conclude with a summary and outlook for future research topics.

Preliminaries, classical RDFS

Syntax. Consider pairwise disjoint alphabets $\mathbf{U}$, $\mathbf{B}$, and $\mathbf{L}$ denoting, respectively, *URI references*, *Blank nodes* and *Literals*.[2] Elements in $\mathbf{UBL}$ (resp. $\mathbf{B}$) are *terms* (resp. *variables*, denoted x, y, z). An *RDF triple* is $\tau = (s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$.[3] We call s the *subject*, p the *predicate*, and o the *object*. A *graph* G is a set of triples, the *universe* of G, $universe(G)$, is the set of elements in $\mathbf{UBL}$ that occur in the triples of G, the *vocabulary* of G, $voc(G)$,

[2] We assume $\mathbf{U}, \mathbf{B}$, and $\mathbf{L}$ fixed, and for ease we will denote unions of these sets simply concatenating their names.

[3] As in (Muñoz, Pérez, Gutierrez 2007) we allow literals for s.

is $universe(G) \cap \mathbf{UL}$. In this work, we rely on a slight variant of a minimal RDFS fragment, called ρdf (Muñoz, Pérez, Gutierrez 2007), that covers essential features of RDFS [4]. ρdf is defined as the following subset of the RDFS vocabulary used as predicates: $\rho\text{df} = \{\mathsf{sp}, \mathsf{sc}, \mathsf{type}, \mathsf{dom}, \mathsf{range}\}$. Informally, (i) (p, sp, q) means that property p is a *subproperty* of property q; (ii) (c, sc, d) means that class c is a *subclass* of class d; (iii) (a, type, b) means that a is of *type* b; (iv) (p, dom, c) means that the *domain* of property p is c; and (v) (p, range, c) means that the *range* of property p is c.

Semantics. An *interpretation* $\mathcal{I}$ over vocabulary V is a tuple $\mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\![\cdot]\!], C[\![\cdot]\!], \cdot^{\mathcal{I}} \rangle$, where $\Delta_R, \Delta_P, \Delta_C, \Delta_L$ are the interpretation domains (*i.e.*, non-empty sets) of $\mathcal{I}$, $P[\![\cdot]\!], C[\![\cdot]\!], \cdot^{\mathcal{I}}$ are the interpretation functions of $\mathcal{I}$, and:

1. Δ_R are the resources (the domain or universe of $\mathcal{I}$);
2. Δ_P are property names (not necessarily disjoint from Δ_R);
3. $\Delta_C \subseteq \Delta_R$ are the classes;
4. $\Delta_L \subseteq \Delta_R$ are the literal values and contains $\mathbf{L} \cap V$;
5. $P[\![\cdot]\!]$ is a function $P[\![\cdot]\!] \colon \Delta_P \to 2^{\Delta_R \times \Delta_R}$;
6. $C[\![\cdot]\!]$ is a function $C[\![\cdot]\!] \colon \Delta_C \to 2^{\Delta_R}$;
7. on $\mathbf{UL}$, $\cdot^{\mathcal{I}}$ is a function $\cdot^{\mathcal{I}} \colon \mathbf{UL} \cap V \to \Delta_R \cup \Delta_P$ such that $\cdot^{\mathcal{I}}$ is the identity for plain literals and $\cdot^{\mathcal{I}} \colon \mathbf{L} \to \Delta_R$;
8. on $\mathbf{B}$, $\cdot^{\mathcal{I}}$ is a function $\cdot^{\mathcal{I}} \colon \mathbf{B} \to \Delta_R$.

Entailment is defined as usual. Intuitively, a triple (s, p, o) in a graph G is true under the interpretation $\mathcal{I}$ if p is interpreted as a property name, s and o are interpreted as resources, and the interpretation of the pair (s, o) belongs to the extension of the property assigned to p. Moreover, blank nodes are treated as skolem constants, *e.g.* the triple (x, p, o) with $x \in \mathbf{B}$ will be true under $\mathcal{I}$ if $x^{\mathcal{I}} = s^{\mathcal{I}}$, for some s such that (s, p, o) is true under $\mathcal{I}$ (cf. also Remark 2 below). So, an interpretation $\mathcal{I}$ is a *model* of a graph G, denoted $\mathcal{I} \models G$, iff $\mathcal{I}$ is an interpretation over the vocabulary $\rho\text{df} \cup universe(G)$, $\mathcal{I}$ satisfies the conditions in Table 1 and makes all triples in G true. We define G *entails* a triple τ under ρdf, denoted $G \models \tau$, iff every model under ρdf of G is also a model under ρdf of τ.

Remark 1 *In (Muñoz, Pérez, Gutierrez 2007), $P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ (resp. $C[\![\mathsf{sc}^{\mathcal{I}}]\!]$) may also be* reflexive *over Δ_P (resp. Δ_C). For ease, we omit this requirement and, thus, do not support an inference such as $G \models (a, \mathsf{sc}, a)$, which anyway are of marginal interest (see (Muñoz, Pérez, Gutierrez 2007) for a more in depth discussion on this issue).*

Remark 2 *We have a slightly different handling of variables, but which is harmless from a query answering point of view. We are not supporting entailments such as $(s, p, z) \models (s, p, y)$ (z, y are variables). The reason is that (i) we have a simpler inference rule schema; and (ii) we are going to focus on query answering and not on graph entailment.*

Remark 3 *In a First-Order Logic (FOL) setting, we may interpret classes as unary predicates, and predicates as binary predicates. Then (i) a subclass relation between class c and d may be encoded as the formula $\forall x.c(x) \Rightarrow d(x)$; (ii) a subproperty relation between property p and q may be encoded as $\forall x \forall y.p(x, y) \Rightarrow q(x, y)$; (iii) domain and range properties may be represented as $\forall x \forall y.p(x, y) \Rightarrow c(x)$ and $\forall x \forall y.p(x, y) \Rightarrow c(y)$; (iv) the transitivity of a property can be represented as $\forall x \forall y \exists z.(p(x, z) \wedge p(z, y)) \Rightarrow p(x, y)$.*

Although this remark is trivial, we will see that it will play an important role in the formalisation of annotated RDFS.

Deductive system. For space reasons, we do not describe here the deductive system as it corresponds exactly to the set of rules 2-5 as in (Muñoz, Pérez, Gutierrez 2007) (which has 7 rules). We rule out the rules handling reflexivity (rules 6-7) and the one dealing with variables (rule 1a), the latter we do not need to answer queries. The notion of *proof* of τ from G, denoted $G \vdash \tau$, is defined as usual and we have:

Proposition 1 ((Muñoz, Pérez, Gutierrez 2007)) *Inference $\vdash$ based on rules 2-5 as of (Muñoz, Pérez, Gutierrez 2007) and applied to our semantics defined above is sound and complete for $\models$, that is, $G \vdash \tau$ iff $G \models \tau$.*

Query Answering. Concerning query answering, we are inspired by (Gutierrez, Hurtado, Mendelzon 2004) and the Logic Programming setting. A *query* is of the rule-like form $q(\bar{\mathbf{x}}) \leftarrow \exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, where $q(\bar{\mathbf{x}})$ is the *head* and $\exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is the *body* of the query, which is a boolean combination (using $\wedge, \vee$) of triples τ_i ($1 \leqslant i \leqslant n$). $\bar{\mathbf{x}}$ is a vector of variables occurring in the body, called the *distinguished variables*, $\bar{\mathbf{y}}$ are so-called *non-distinguished variables* and are distinct from the variables in $\bar{\mathbf{x}}$, each variable occurring in τ_i is either a distinguished or a non-distinguished variable. If clear from the context, we may omit the existential quantification $\exists \bar{\mathbf{y}}$. For instance, the query $q(x) \leftarrow (y, created, x) \wedge (y, \mathsf{type}, Italian) \wedge (x, exhibited, Uffizi)$ has intended meaning to retrieve all the artifacts x created by Italian artists y, being exhibited at Uffizi Gallery.

Given a graph G, a query $q(\bar{\mathbf{x}}) \leftarrow \exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, and a vector $\bar{\mathbf{t}}$ of terms in $universe(G)$, we say that $q(\bar{\mathbf{t}})$ is *entailed* by G, denoted $G \models q(\bar{\mathbf{t}})$, iff in any model $\mathcal{I}$ of G, there is a vector $\bar{\mathbf{t}}'$ of terms in $universe(G)$ such that $\mathcal{I}$ is a model of $\varphi(\bar{\mathbf{t}}, \bar{\mathbf{t}}')$. If $G \models q(\bar{\mathbf{t}})$ then $\bar{\mathbf{t}}$ is called an *answer* to q. The *answer set* of q w.r.t. G is defined as $ans(G, q) = \{\bar{\mathbf{t}} \mid G \models q(\bar{\mathbf{t}})\}$.

We next show how to compute the answer set. The *closure* of a graph G is defined as $cl(G) = \{\tau \mid G \vdash \tau\}$. Note that the size of the closure of G is polynomial in the size of G and that the closure is *unique*. Now, the following holds:

Proposition 2 *Given a graph G, $\bar{\mathbf{t}}$ is an* answer *to q iff $\exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{t}}, \bar{\mathbf{y}})$ is true in the closure of G.* [5]

Therefore, we have a simple method to determine $ans(G, q)$. Compute the closure $cl(G)$ and store it into a database, *e.g.*, using the method (Ianni et al. 2009). It is easily verified that any query can be mapped into an SQL query over the underlying database schema. Hence, $ans(G, q)$ is determined by issuing such an SQL query to the database.

RDFS with Annotations

Syntax. Our approach is to extend triples with annotations, where an annotation is taken from a specific domain. Roughly, our approach is somewhat related to the annotated logic programming framework (Kifer, Subrahmanian 1992).

[4] ρdf is considered as a good candidate to address RDFS extensions, allowing focusing on the main problem. Extension to full RDFS rulesets is easy.

[5] The evaluation of the truth of a formula is as usual.

Simple:	1. for each $(s,p,o) \in G, p^{\mathcal{I}} \in \Delta_P$ and $(s^{\mathcal{I}}, o^{\mathcal{I}}) \in P[\![p^{\mathcal{I}}]\!]$;	**Typing I:**	1. $x \in C[\![c]\!]$ iff $(x,c) \in P[\![\mathsf{type}^{\mathcal{I}}]\!]$; 2. if $(p,c) \in P[\![\mathsf{dom}^{\mathcal{I}}]\!]$ and $(x,y) \in P[\![p]\!]$ then $x \in C[\![c]\!]$; 3. if $(p,c) \in P[\![\mathsf{range}^{\mathcal{I}}]\!]$ and $(x,y) \in P[\![p]\!]$ then $y \in C[\![c]\!]$;
Subproperty:	1. $P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ is transitive over Δ_P; 2. if $(p,q) \in P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ then $p,q \in \Delta_P$ and $P[\![p]\!] \subseteq P[\![q]\!]$;	**Typing II:**	1. for each $\mathsf{e} \in \rho\mathrm{df}$, $\mathsf{e}^{\mathcal{I}} \in \Delta_P$ 2. if $(p,c) \in P[\![\mathsf{dom}^{\mathcal{I}}]\!]$ then $p \in \Delta_P$ and $c \in \Delta_C$ 3. if $(p,c) \in P[\![\mathsf{range}^{\mathcal{I}}]\!]$ then $p \in \Delta_P$ and $c \in \Delta_C$ 4. if $(x,c) \in P[\![\mathsf{type}^{\mathcal{I}}]\!]$ then $c \in \Delta_C$
Subclass:	1. $P[\![\mathsf{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C; 2. if $(c,d) \in P[\![\mathsf{sc}^{\mathcal{I}}]\!]$ then $c,d \in \Delta_C$ and $C[\![c]\!] \subseteq C[\![d]\!]$;		

Table 1: The conditions for classical interpretations.

An *annotated triple* is an expression $\tau : v$, where τ is a triple and v is an *annotation term* (defined below). An *annotated graph* is a finite set of annotated triples. The intended semantics of annotated triples depends of course on the meaning we associate to the annotation terms. For instance, in a temporal setting (Gutierrez, Hurtado, Vaisman 2007), $(AlainProst, \mathsf{type}, F1Driver) : [1980, 1991]$ has intended meaning "Alain Prost was a Formula One driver during 1980-1991", while in the fuzzy setting (Straccia 2009) $(audiTT, \mathsf{type}, SportsCar) : 0.8$ has intended meaning "AudiTT is a sports car to degree not less than 0.8".

RDFS Annotation Domains. To start with, let us consider a lattice $\langle L, \preceq \rangle$. Elements in L are our annotation terms. *E.g.*, in a fuzzy setting, $L = [0,1]$, while in a typical temporal setting, L may be time points or time intervals. The order $\preceq$ is used to express redundant/entailed/subsumed information. For instance, for temporal intervals, an annotated triple $\langle s,p,o \rangle : [2,6]$ entails $\langle s,p,o \rangle : [3,4]$, as $[3,4] \subseteq [2,6]$ (here, $\subseteq$ plays the role of $\preceq$).

In our annotation framework, an interpretation will map statements to elements of the annotation domain. Hence, we have to guarantee that the formulae in Remark 3, are properly interpreted. To this end, we will use an algebraic structure that is well-known for Many-Valued FOL (Hájek 1998). In light of Remark 3, our choice may look less surprising as it may be at first sight. We say that an *annotation domain* for RDFS is an algebraic structure

$$D = \langle L, \preceq, \wedge, \vee, \otimes, \Rightarrow, \bot, \top \rangle$$

such that (i) $\langle L, \preceq, \wedge, \vee, \bot, \top \rangle$ is a lattice, with bottom and top elements $\bot$ and $\top$, meet and join operator $\wedge$ and $\vee$, respectively; (ii) $\otimes$ is a so-called t-norm (Hájek 1998), *i.e.* is commutative, associative, monotonic, and $\top$ acts as identity element (for any $v \in L$, $v \otimes \top = v$); and (iii) $\Rightarrow$ is the so-called residuum of $\otimes$, *i.e.* $v_1 \Rightarrow v_2 = \sup\{v \mid v_1 \otimes v \preceq v_2\}$.

Remark 4 *We will use $\vee$ to combine information about the same statement.* E.g.*, in temporal logic, from $\tau : [2,6]$ and $\tau : [3,8]$, we will infer $\tau : [2,8]$, as $[2,8] = [2,6] \cup [3,8]$; here, $\cup$ plays the role of $\vee$. In the fuzzy context, from $\tau : 0.7$ and $\tau : 0.6$, we will infer $\tau : 0.7$, as $0.7 = \max(0.7, 0.6)$ (here, $\max$ plays the role of $\vee$).*

Remark 5 *We will use the t-norm $\otimes$ to model the "conjunction" of information. In fact, a t-norm is a generalisation of boolean conjunction to the many-valued case. For instance, on interval-valued temporal logic, from $\langle a, \mathsf{sc}, b \rangle : [2,6]$ and $\langle b, \mathsf{sc}, c \rangle : [3,8]$, we will infer $\langle a, \mathsf{sc}, c \rangle : [3,6]$, as $[3,6] = [2,6] \cap [3,8]$; here, $\cap$ plays the role of $\otimes$.*[6] *In the fuzzy context, one may chose any t-norm,* e.g. *product, and, thus, from $\langle a, \mathsf{sc}, b \rangle : 0.7$ and $\langle b, \mathsf{sc}, c \rangle : 0.6$, we will infer $\langle a, \mathsf{sc}, c \rangle : 0.42$, as $0.42 = 0.7 \cdot 0.6$ (here, $\cdot$ plays the role of $\otimes$). Please note that in this latter case, $\wedge$ and $\otimes$ do not coincide as $\wedge$ is $\min$ (i.e., the so-called Gödel t-norm).*

[6] As we will see, $\vee$ and $\otimes$ may be more involved.

Remark 6 *One may be tempted to consider other combination functions than $\vee$ such as so-called s-norms ($\oplus$) (Hájek 1998) that are used to model "disjunction".* E.g.*, in the fuzzy context one may think of using the probabilistic sum $n \oplus m = n + m - n \cdot m$. So,* e.g.*, from $\tau : 0.7$ and $\tau : 0.6$, we may infer $\tau : 0.88$, as $0.88 = 0.7 + 0.6 - 0.7 \cdot 0.6$. However, we are not using them as then the termination of our calculus is not guaranteed if schema cycles are involved in a graph, such as $\langle a, \mathsf{sc}, b \rangle : 0.7$, $\langle b, \mathsf{sc}, a \rangle : 0.6$. The interested reader may verify that after an infinite number of inference steps we will infer* e.g. *$\langle a, \mathsf{sc}, b \rangle : 1.0$. Using $\vee$ (which is the so-called Gödel s-norm) will guarantee termination. In principle, we may workout easily the case using $\oplus$ in place of $\vee$ as well, provided we restricted graphs to be "schema acyclic", or assume that L is finite, to guarantee termination of the inference.*

Remark 7 *The function $\Rightarrow$, which is defined in terms of $\otimes$ only, will be used to model the "implication" that occurs in Remark 3, and is called* r-implication *(which stands for* residuated implication*). It is acknowledged that an r-implication, which is a generalisation of boolean implication to the many-valued case, has many of the properties that an implication functions should have, namely, is monotone in the first argument, is antitone in the second argument, $v \Rightarrow v'$ evaluates to $\top$ iff $v \preceq v'$ and $\top \Rightarrow \bot = \bot$. A feature of r-implications is that it supports modus ponens inference schemas,* i.e.

$$a \succeq v \text{ and } a \Rightarrow b \succeq v' \quad \textit{imply} \quad b \succeq v \otimes v' \tag{1}$$

$$a \Rightarrow b \succeq v \text{ and } b \Rightarrow c \succeq v' \quad \textit{imply} \quad a \Rightarrow c \succeq v \otimes v' \tag{2}$$

(these are the main inference patterns we will rely on).

Remark 8 *We will not use explicitly $\wedge$. However, $\wedge$ is implicitly present on the structure as it can be defined in terms of $\otimes$ and $\Rightarrow$ as $v \wedge v' = v \otimes (v \Rightarrow v')$.*

By the above remarks, we may represent an annotation domain succinctly as an algebraic structure $D = \langle L, \preceq, \otimes, \bot, \top \rangle$ in which $\langle L, \preceq, \top, \bot \rangle$ is a bounded lattice and $\otimes$ is a t-norm. Note that for domain $D_{01} = \langle \{0,1\}, \leqslant, \min, 0, 1 \rangle$, annotated RDFS will turn out to be the same as classical RDFS. Finally, note also that in order to build an annotation domain, one has to (i) determine the set of annotation terms L, the partial order $\preceq$, identify the top and bottom elements and guarantee that we obtain a bounded lattice; and (ii) define a suitable t-norm $\otimes$ that acts as "conjunction" function, to support the intended inference over schema axioms, such as "from $\langle a, \mathsf{sc}, b \rangle : v$ and $\langle b, \mathsf{sc}, c \rangle : v'$ infer $\langle a, \mathsf{sc}, c \rangle : v \otimes v'$" (see inference pattern (2)).

Semantics. Fix a domain $D = \langle L, \preceq, \otimes, \bot, \top \rangle$. Informally, an interpretation $\mathcal{I}$ will assign to a triple τ an ele-

ment of the annotation domain $v \in L$, dictating that under $\mathcal{I}$, the annotation of τ is greater or equal than v. Formally, an *annotated interpretation* $\mathcal{I}$ over a vocabulary V is a tuple $\mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\![\cdot]\!], C[\![\cdot]\!], \cdot^{\mathcal{I}} \rangle$, where $\Delta_R, \Delta_P, \Delta_C, \Delta_L$ and $\cdot^{\mathcal{I}}$ are exactly as for the classical case (items 1-4, 7-8), while $P[\![\cdot]\!], C[\![\cdot]\!]$ are interpretation functions for which items 5 and 6 are changed as follows:

5'. $P[\![\cdot]\!]$ is a function $P[\![p]\!] : \Delta_R \times \Delta_R \to L$ assigning an annotation term to each pair of resources;

6'. $C[\![\cdot]\!]$ is a function $C[\![c]\!] : \Delta_R \to L$ assigning an annotation term to every resource;

It is immediate to see that the classical setting is as the case in which the annotation domain is D_{01}, *i.e.* $L = \{0, 1\}$.

Now, consider a set $\Delta \subseteq \Delta_R \cup \Delta_P$, we say that a function $p\colon \Delta \times \Delta \to L$ is $\sup - \otimes$ *transitive* (or simply *transitive*) over Δ iff for all $x, y \in \Delta$, $\sup_{z \in \Delta} \{p(x, z) \otimes p(z, y)\} \preceq p(x, y)$. Please note the relationship to case 4 in Remark 3 ($\otimes$ plays the role of conjunction, sup plays the role of the existential over Δ and the implication evaluates to $\top$ iff the antecedent is $\preceq$ than the succedent, see Remark 7).

Entailment is defined accordingly. Intuitively, a triple $(s, p, o)\colon v$ is satisfied by $\mathcal{I}$ if (s, o) belongs to the extension of p to a "wider" extent than v. Formally, an annotated interpretation $\mathcal{I}$ over the vocabulary $\rho\text{df} \cup universe(G)$ is a *model* of an annotated graph G under ρdf, denoted $\mathcal{I} \models G$, iff $\mathcal{I}$ satisfies the conditions in Table 2. Please note how now the conditions follow immediately from the classical RDFS setting (Table 1) by relying on Remark 3 (the universal quantifier is interpreted as inf), *e.g.* the condition $P[\![\mathsf{sc}^{\mathcal{I}}]\!](c, d) = \inf_{x \in \Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$ follows immediately from $\forall x.c(x) \Rightarrow d(x)$. Finally, we define *$G$ entails $\tau\colon v$* under ρdf, denoted $G \models \tau\colon v$, iff every annotated model under ρdf of G is also a model under ρdf of $\tau\colon v$. As for the crisp case, it can be shown that

Proposition 3 *Any annotated RDFS graph has a model.*

Therefore, we do not have to care about consistency.

Deductive system. An important feature of our framework is that we are able to provide a deductive system in the style of the one for classical RDFS. Moreover, *the schemata of the rules are the same for any annotation domain* (only support for the domain dependent $\otimes$ and $\vee$ operations has to be provided) and, thus, are amenable to an easy implementation on top of existing systems. The rules (see Table 3) are arranged in groups that capture the semantic conditions of models, A, B, C, X and Y are meta-variables representing elements in **UBL**. The notion of *proof* is as usual.

Proposition 4 (Soundness and completeness) *For an annotated graph G, (1) if $G \vdash \tau : v$ then $G \models \tau : v$ and (2) if $G \models \tau : v$ then there is $v' \succeq v$ with $G \vdash \tau : v'$.*

Finally, like for the classical case, the *closure* is defined as $cl(G) = \{\tau\colon v \mid G \vdash \tau\colon v\}$, the size of the closure of G is polynomial in $|G|$ and can be computed in polynomial time, provided that the computational complexity of operations $\otimes$ and $\vee$ are polynomially bounded (from a computational complexity point of view, it is as for the classical case, plus the cost of the operations $\otimes$ and $\vee$ in L).

Query Answering. Informally, queries are as for the classical case where triples are replaced with annotated triples in which *annotation variables* (taken from an appropriate alphabet and denoted V) may occur.

Example 1 *The following temporal information where labels mark intervals in years (the exact meaning will be addressed later on) can be extracted from Wikipedia:*[7]

$(AlainProst, \mathsf{type}, RenaultF1Driver)\colon [1981, 1983]$
$(AlainProst, \mathsf{type}, McLarenF1Driver)\colon \{[1980], [1984, 1989]\}$
$(AlainProst, \mathsf{type}, FerrariF1Driver)\colon [1990, 1991]$
$(AlainProst, \mathsf{type}, WilliamsF1Driver)\colon [1993]$
$(McLarenF1Driver, \mathsf{sc}, F1Driver)\colon [1966, 2010]$
$(RenaultF1Driver, \mathsf{sc}, F1Driver)\colon \{[1977, 1985], [2001, 2009]\}$
$(FerrariF1Driver, \mathsf{sc}, F1Driver)\colon [1950, 2010]$
$(WilliamsF1Driver, \mathsf{sc}, F1Driver)\colon [1976, 2010]$
$(F1Driver, \mathsf{sc}, SportsCarDriver)\colon \top$

The query asking for sports car drivers between 1975 and 1985 and the temporal term at which this was true $q(x, V) \leftarrow (x, \mathsf{type}, SportsCarDriver)\colon V \ \wedge \ (V, \preceq, [1975, 1985])$ will get the answer $\langle AlainProst, [1980, 1985] \rangle$.

Formally, an *annotated query* is of the form $q(\bar{\mathbf{x}}, \bar{\mathbf{V}}) \leftarrow \exists \bar{\mathbf{y}} \exists \mathbf{V}'.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{V}}, \bar{\mathbf{y}}, \bar{\mathbf{V}}')$ in which $\varphi(\bar{\mathbf{x}}, \bar{\mathbf{V}}, \bar{\mathbf{y}}, \bar{\mathbf{V}}')$ is a boolean combination of annotated triples, $\bar{\mathbf{x}}$ and $\bar{\mathbf{V}}$ are the distinguished variables, $\bar{\mathbf{y}}$ and $\bar{\mathbf{V}}'$ are the vectors of *non-distinguished variables* (existential quantified variables), and $\bar{\mathbf{x}}$, $\bar{\mathbf{V}}$, $\bar{\mathbf{y}}$ and $\bar{\mathbf{V}}'$ are pairwise disjoint. The query head contains at least one variable.

Given an annotated graph G, a query $q(\bar{\mathbf{x}}, \bar{\mathbf{V}}) \leftarrow \exists \bar{\mathbf{y}} \exists \mathbf{V}'.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{V}}, \bar{\mathbf{y}}, \bar{\mathbf{V}}')$, a vector $\bar{\mathbf{t}}$ of terms in $universe(G)$ and a vector $\bar{\mathbf{v}}$ of annotated terms in L, we say that $q(\bar{\mathbf{t}}, \bar{\mathbf{v}})$ is *entailed* by G, denoted $G \models q(\bar{\mathbf{t}}, \bar{\mathbf{v}})$, iff in any model $\mathcal{I}$ of G, there is a vector $\bar{\mathbf{t}}'$ of terms in $universe(G)$ and a vector $\bar{\mathbf{v}}'$ of annotation terms in L such that $\mathcal{I}$ is a model of $\varphi(\bar{\mathbf{t}}, \bar{\mathbf{v}}, \bar{\mathbf{t}}', \bar{\mathbf{v}}')$. If $G \models q(\bar{\mathbf{t}}, \bar{\mathbf{v}})$ then $\langle \bar{\mathbf{t}}, \bar{\mathbf{v}} \rangle$ is called an *answer* to q. The *answer set* of q w.r.t. G is ($\preceq$ extends to vectors point-wise)

$$ans(G, q) = \{\langle \bar{\mathbf{t}}, \bar{\mathbf{v}} \rangle \mid G \models q(\bar{\mathbf{t}}, \bar{\mathbf{v}}) \text{ and for any } \bar{\mathbf{v}}' \neq \bar{\mathbf{v}} \text{ such that } G \models q(\bar{\mathbf{t}}, \bar{\mathbf{v}}'), \bar{\mathbf{v}}' \preceq \bar{\mathbf{v}} \text{ holds}\} .$$

That is, for any tuple $\bar{\mathbf{t}}$, the vector of annotation terms $\bar{\mathbf{v}}$ is as large as possible. This is to avoid that redundant/subsumed answers occur in the answer set. The following holds:

Proposition 5 *Given a graph G, $\langle \bar{\mathbf{t}}, \bar{\mathbf{v}} \rangle$ is an* answer *to q iff $\exists \bar{\mathbf{y}} \exists \mathbf{V}'.\varphi(\bar{\mathbf{t}}, \bar{\mathbf{v}}, \bar{\mathbf{y}}, \bar{\mathbf{V}}')$ is true in the closure of G.*

Therefore, we may devise a similar query answering method as for the crisp case by computing the closure, store it into a database and then using SQL queries.

Two Applications: Fuzzy and Temporal RDFS

To demonstrate the power of our approach, we illustrate its application to two domains: fuzzy RDFS (Straccia 2009) and temporal RDFS (Gutierrez, Hurtado, Vaisman 2007).

The fuzzy Domain. To model fuzzy RDFS is easy: the annotation domain is $D_{[0,1]} = \langle [0, 1], \leqslant, \otimes, 0, 1 \rangle$ where $\otimes$ is any t-norm on $[0, 1]$ and $\vee$ is max.

Example 2 *Under the product t-norm $\otimes$, given $(audiTT, \mathsf{type}, SportsCar)\colon 0.8$, $(BMWM3, \mathsf{type}, SportsCar)\colon 0.9$, and $(SportsCar, \mathsf{sc}, ExpensiveCar)\colon 0.9$, the answers to the query $q(x, V) \leftarrow (x, \mathsf{type}, ExpensiveCar)\colon V$ ("retrieve expensive cars"), are $\langle audiTT, 0.72 \rangle$ and $\langle BMWM3, 0.81 \rangle$.*

[7] cf. *e.g.* `http://en.wikipedia.org/wiki/Alain_Prost`

Simple:	1. $(s,p,o)\colon v \in G$ implies $p^{\mathcal{I}} \in \Delta_P$ and $P[\![p^{\mathcal{I}}]\!](s^{\mathcal{I}}, o^{\mathcal{I}}) \succeq v$	**Typing I:**	1. $C[\![c]\!](x) = P[\![\mathsf{type}^{\mathcal{I}}]\!](x,c)$ 2. $P[\![\mathsf{dom}^{\mathcal{I}}]\!](p,c) = \inf_{(x,y)\in\Delta_R\times\Delta_R} P[\![p]\!](x,y) \Rightarrow C[\![c]\!](x)$ 3. $P[\![\mathsf{range}^{\mathcal{I}}]\!](p,c) = \inf_{(x,y)\in\Delta_R\times\Delta_R} P[\![p]\!](x,y) \Rightarrow C[\![c]\!](y)$
Subproperty:	1. $P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ is transitive over Δ_P 2. $P[\![\mathsf{sp}^{\mathcal{I}}]\!](p,q) = \inf_{(x,y)\in\Delta_R\times\Delta_R} P[\![p]\!](x,y) \Rightarrow P[\![q]\!](x,y)$	**Typing II:**	1. For each $\mathsf{e} \in \rho\mathsf{df}$, $\mathsf{e}^{\mathcal{I}} \in \Delta_P$ 2. $P[\![\mathsf{dom}^{\mathcal{I}}]\!](p,c)$ is defined only for $p \in \Delta_P$ and $c \in \Delta_C$ 3. $P[\![\mathsf{range}^{\mathcal{I}}]\!](p,c)$ is defined only for $p \in \Delta_P$ and $c \in \Delta_C$ 4. $P[\![\mathsf{type}^{\mathcal{I}}]\!](s,c)$ is defined only for $c \in \Delta_C$
Subclass:	1. $P[\![\mathsf{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C 2. $P[\![\mathsf{sc}^{\mathcal{I}}]\!](c,d) = \inf_{x\in\Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$		

Table 2: The conditions for annotated interpretations.

1. **Subproperty:** (a) $\dfrac{(A,\mathsf{sp},B)\colon v_1, (B,\mathsf{sp},C)\colon v_2}{(A,\mathsf{sp},C)\colon v_1 \otimes v_2}$ (b) $\dfrac{(A,\mathsf{sp},B)\colon v_1, (X,A,Y)\colon v_2}{(X,B,Y)\colon v_1 \otimes v_2}$

2. **Subclass:** (a) $\dfrac{(A,\mathsf{sc},B)\colon v_1, (B,\mathsf{sc},C)\colon v_2}{(A,\mathsf{sc},C)\colon v_1 \otimes v_2}$ (b) $\dfrac{(A,\mathsf{sc},B)\colon v_1, (X,\mathsf{type},A)\colon v_2}{(X,\mathsf{type},B)\colon v_1 \otimes v_2}$

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

4. **Implicit Typing:** (a) $\dfrac{(A,\mathsf{dom},B)\colon v_1, (C,\mathsf{sp},A)\colon v_2, (X,C,Y)\colon v_3}{(X,\mathsf{type},B)\colon v_1 \otimes v_2 \otimes v_3}$ (b) $\dfrac{(A,\mathsf{range},B)\colon v_1, (C,\mathsf{sp},A)\colon v_2, (X,C,Y)\colon v_3}{(Y,\mathsf{type},B)\colon v_1 \otimes v_2 \otimes v_3}$

5. **Generalisation:** $\dfrac{(X,A,Y)\colon v_1, (X,A,Y)\colon v_2}{(X,A,Y)\colon v_1 \vee v_2}$

Table 3: Inference rules for annotated RDFS.

Note that in the same manner we may build a trust annotation domain modelling the reliability of a triple.

The Temporal Domain. Modelling the temporal domain turns out to be more involved than the fuzzy case. To start with, *time points* are elements of $\mathbb{P} = \mathbb{Z} \cup \{-\infty, +\infty\}$, where $\mathbb{Z}$ are the integers. A *temporal interval* is a non-empty interval $[\alpha_1, \alpha_2]$, where α_i are time points. An empty interval is denoted as $\emptyset$. We define a partial order on intervals as $I_1 \leqslant I_2$ iff $I_1 \subseteq I_2$. The intuition here is that if a triple is true at time points in I_2 and $I_1 \leqslant I_2$ then, in particular, it is true at any time point in I_1. Now, apparently the set of intervals would be a candidate for L, which however is not the case. The reason is that, *e.g.*, in order to represent the upper bound interval of $\tau\colon [1,5]$ and $\tau\colon [8,9]$ we rather need the disjoint union of intervals, denoted $\{[1,5],[8,9]\}$, meaning that a triple is true both in the former as well as in the latter interval. Formally, we say that a finite set of intervals t is *maximal* iff the intervals in t are pairwise disjoint and there are no two *adjacent* intervals in it, where $[\alpha_1, \alpha_2]$ and $[\alpha_3, \alpha_4]$ are adjacent iff $\alpha_3 = \alpha_2 + 1$. Now, we define L as (where $\bot = \{\emptyset\}, \top = \{[-\infty, +\infty]\}$)

$$L \quad = \quad \{t \mid t \text{ is a finite and maximal set of temporal intervals } \} \cup \{\bot, \top\} \ .$$

Therefore, a *temporal term* is an element $t \in L$, *i.e.* a set of pairwise disjoint time intervals. We allow to write $[\alpha]$ as a shorthand for $[\alpha, \alpha]$, $\tau\colon \alpha$ as a shorthand of $\tau\colon \{[\alpha]\}$ and $\tau\colon [\alpha, \alpha']$ as a shorthand of $\tau\colon \{[\alpha, \alpha']\}$. Furthermore, on L we define the following partial order:

$$t_1 \preceq t_2 \text{ iff } \forall I_1 \in t_1 \exists I_2 \in t_2, \text{ such that } I_1 \leqslant I_2 \ .$$

Please note that $\preceq$ is the Hoare order on power sets (Abramsky, Jung 1994), which is a pre-order. For the anti-symmetry property, assume that $t_1 \preceq t_2$ and $t_2 \preceq t_1$: so for $I_1 \in t_1$, there is $I_2 \in t_2$ for which there is $I_3 \in t_1$ such that $I_1 \subseteq I_2 \subseteq I_3$. But, t_1 is maximal and, thus, $I_1 = I_3 = I_2$. So, $t_1 = t_2$ and, thus, $\preceq$ is a partial order. Similarly as for time intervals, the intuition for $\preceq$ is that if a triple is true at time points in intervals in t_2 and $t_1 \preceq t_2$, then, in particular, it is true at any time point in intervals in t_1. Essentially, if $t_1 \preceq t_2$ then a temporal triple $\tau_2\colon t_2$ is true to a larger "temporal extent" than the temporal triple $\tau_1\colon t_1$. It can also be verified that $\langle L, \preceq, \bot, \top\rangle$ is a bounded lattice. Indeed, to what concerns us, the partial order $\preceq$ induces the following join ($\vee$) operation on L. Intuitively, if a triple is true at t_1 and also true at t_2 then it will be true also for time points specified by $t_1 \vee t_2$ (a kind of union of time points). *E.g.*, if $\tau\colon \{[2,5],[8,12]\}$ and $\tau\colon \{[4,6],[9,15]\}$ are true then we expect that this is the same as saying that $\tau\colon \{[2,6],[8,15]\}$ is true. The join operator will be defined in such way that $\{[2,5],[8,12]\} \vee \{[4,6],[9,15]\} = \{[2,6],[8,15]\}$. Operationally, this means that $t_1 \vee t_2$ will be obtained as follows: (i) take the union of the sets of intervals $t = t_1 \cup t_2$; and (ii) join overlapping intervals in t until no more overlapping intervals can be obtained. Formally,

$$t_1 \vee t_2 = \inf\{t \mid t \succeq t_i, i = 1,2\} \ .$$

It remains to define the t-norm $\otimes$ over sets of intervals. Intuitively, we would like to support inferences such as "from $\langle a, \mathsf{sc}, b\rangle\colon \{[2,5],[8,12]\}$ and $\langle b, \mathsf{sc}, c\rangle\colon \{[4,6],[9,15]\}$ infer $\langle a, \mathsf{sc}, b\rangle\colon \{[4,5],[9,12]\}$", where $\{[2,5],[8,12]\} \otimes \{[4,6],[9,15]\} = \{[4,5],[9,12]\}$. We get it by means of

$$t_1 \otimes t_2 = \sup\{t \mid t \preceq t_i, i = 1,2\} \ .$$

Note that here the t-norm used for modelling "conjunction" coincides with the lattice meet operator $\wedge$ (Gödel's t-norm) and, thus, the temporal domain $D_T = \langle L, \preceq, \otimes, \bot, \top\rangle$ coincides with the bounded lattice $D_T = \langle L, \preceq, \bot, \top\rangle$.

Example 3 *From Example 1 we can infer by multiple application of the rules 2.(b) and 5. from Table 3 that* $(AlainProst, \mathsf{type}, SportsCarDriver)\colon \{[1980,1991],[1993]\}$, *where,* e.g. *to infer* $F1Driver$ *membership during Alain's Renault period we calculate* $\{[1977,1985],[2001,2009]\} \otimes \{[1981,1983]\} = \{[1981,1983]\}$, *for* $SportsCarDriver$ *membership in that period we calculate* $\{[1981,1983]\} \otimes \top = \{[1981,1983]\}$, *and finally* $\{[1981,1983]\} \vee \{[1980],[1984,1989]\} \vee \{[1990,1991] \vee \{[1993]\} = \{[1980,1991],[1993]\}$.

(Gutierrez, Hurtado, Vaisman 2007) describe some further features such as a "Now" time point (which is just a defined time point in D_T) and anonymous time points, allowing to state that a triple is true at some point. Adding anonymous time points would require us to extend the lattice by appropriate operators, *e.g.* $[4,T] \vee [T,8] = [4,8]$ (where T is an anonymous time point), etc. We do not address this in detail.

Further Considerations

Extensions to multiple domains. Another nice feature of our approach is that it becomes apparent that we may easily combine multiple domains, such as annotating triples

with a temporal term, truth degree, degree of trust, etc. In general, assuming having domains $D_1, \ldots, D_n$, where $D_i = \langle L_i, \preceq_i, \otimes_i, \bot_i, \top_i \rangle$, we may build the domain $D = D_1 \times \ldots \times D_n = \langle L, \preceq, \otimes, \bot, \top \rangle$, where $L = L_1 \times \ldots \times L_n$, $\bot = \langle \bot_1, \ldots, \bot_n \rangle$, $\top = \langle \top_1, \ldots, \top_n \rangle$ and the partial order, meet, join and t-norm operations $\preceq, \wedge, \vee$ and $\otimes$ are extended pointwise to L, *e.g.* $\langle v_1, \ldots, v_n \rangle \otimes \langle v'_1, \ldots, v_n \rangle' = \langle v_1 \otimes v'_1, \ldots, v_n \otimes v'_n \rangle$. For instance,

$(SocialPoliticalScenario, \mathsf{type}, Dangerous) \colon \langle [1975, 1983], 0.8, 0.6 \rangle$

may indicate that the social-political scenario during 1975-1983 in some country has been considered dangerous to degree 0.8 and the degree of trust we have for this statement is 0.6 (here we combine a temporal domain, a fuzzy domain and a domain to represent trust — the latter may be defined similarly as the fuzzy one). The interesting point of our approach is that the rules of the deductive systems need not to be changed as well as the query answering mechanism (except to provide the support to compute $\otimes$ and $\vee$).

Meaning of Classical Triples. While merging annotated and non-annotated triples, a meaning to non-annotated triples has to be given, which depends on the annotation domain. For instance, in the fuzzy case, a triple τ is considered to have degree of truth one and, thus, corresponds to the annotated triple $\tau \colon 1$. While for the fuzzy case the choice is evident, less obvious is the case for the temporal domain. We might, likewise, consider classical triples be annotated with the top of the lattice, i.e. $\tau \colon \top = \tau \colon [-\infty, \infty]$, others (Gutierrez, Hurtado, Vaisman 2007) propose that a classical triple τ may be interpreted as the annotated triple $\tau \colon [-\infty, \mathsf{Now}]$. One may also think of associating a time span $[n, m]$ to a whole classical RDFS graph G and, thus, any triple $\tau \in G$ is interpreted as $\tau \colon [n, m]$. In the combination of annotated triples from different domains, one might find our suggestion to go uniformly with $\top_i$ in the absence of an annotation for domain i appealing since it fits nicely with the idea of combining arbitrarily annotated triples from various domains as sketched above. In any case, such considerations are orthogonal to our framework and domain/application dependent.

Dealing with SPARQL. The boolean expressions allowed in our query language introduced so far allows merely for unions of conjunctive queries. Especially readers familiar with languages such as SQL and SPARQL may appreciate the ability to pose more complex queries including built-in predicates to filter solutions, advanced features such as negation, or aggregates to name a few.

A full elaboration of such extensions of our query language is out of the scope of this paper, but we remark that the striking resemblance of our query language with Datalog is a likely stepping stone for future extensions in this direction: in previous work (Ianni et al. 2009), query answering over RDFS has already been extended to full SPARQL by translations to Datalog. The two missing features in our current query language to enable an analogous translation for annotated SPARQL queries are stratified negation as failure and built-in predicates, the extension for both of which is straightforward in our framework. We leave a concrete elaboration of the syntax (which could be based on former suggestions for temporal/annotated SPARQL such as (Tappolet, Bernstein 2009)) to future work. We are optimistic that features such as aggregate functions currently under elaboration in W3C's SPARQL 1.1working group can also be integrated by relying on, *e.g.*, (Polleres, Scharffe, Schindlauer 2007).

Summary and Conclusion

We have generalised different annotation frameworks for RDF towards RDFS reasoning and query answering. Our framework nicely extends the classical case of RDFS with the features of different annotation domains, such as temporality, fuzzyness, trust, etc. and particularly making these arbitrarily combinable under a common, extensible reasoning framework. A Prolog implementation based on constraint logic programming techniques is ready to play with for the interested reader [8] along with the examples given in this paper. We plan to put more efforts in implementing this approach in a scalable manner and field-test it with temporally annotated data extracted from the Web; here, Linked data sources such as wikipedia/DBpedia may serve as a starting point, which already contain a lot of structured, temporal information amenable to RDF extractors, combined with – for instance – trust values computed by adequate ranking algorithms, such as *e.g.* (Harth, Kinsella, Decker 2009).

Acknowledgements. We thank Antoine Zimmermann for comments. This work was supported by Science Foundation Ireland Grant No. SFI/08/CE/I1380 (Líon-2) and the EU COST Action IC0801 "Agreement Technologies".

References

Abramsky, S., Jung, A. 1994. Domain theory. *Handbook of Logic in Computer Science Volume 3*, 1–168. Oxford University Press.

Dividino, R. Q.; Sizov, S.; Staab, S.; Schueler, B. 2009. Querying for provenance, trust, uncertainty and other meta knowledge in RDF. *JWS* 7(3):204–219, Elsevier.

Gutierrez, C.; Hurtado, C.; Mendelzon, A. O. 2004. Foundations of semantic web databases. *PODS '04*, 95–106. ACM.

Gutierrez, C.; Hurtado, C. A.; Vaisman, A. A. 2007. Introducing Time into RDF. *IEEE Trans. Knowl. Data Eng.* 19(2):207–218. IEEE Press.

Hájek, P. 1998. *Metamathematics of Fuzzy Logic*. Kluwer.

Harth, A.; Kinsella, S.; Decker, S. 2009. Using naming authority to rank data and ontologies for web search. *ISWC '09*, nr. 5823 of *LNCS*, 277–292. Springer.

Hartig, O. 2009. Querying trust in RDF data with tSPARQL. *ESWC '09*, nr. 5554 of *LNCS*, 5–20. Springer.

Ianni, G.; Krennwallner, T.; Martello, A.; Polleres, A. 2009. Dynamic querying of mass-storage RDF data with rule-based entailment regimes. *ISWC '09*, nr. 5823 of *LNCS*, 310–327. Springer.

Kifer, M., Subrahmanian, V. 1992. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming* 12:335–367.

Muñoz, S.; Pérez, J.; Gutierrez, C. 2007. Minimal deductive systems for RDF. *ESWC '07*, 53–67. Springer.

Polleres, A.; Scharffe, F.; Schindlauer, R. 2007. SPARQL++ for mapping between RDF vocabularies. *ODBASE '07)*, nr. 4803 of *LNCS*, 878–896. Springer.

Pugliese, A.; Udrea, O.; Subrahmanian, V. S. 2008. Scaling RDF with time. *WWW '08*, 605–614. ACM.

Straccia, U. 2009. A minimal deductive system for general fuzzy RDF. *RR '09*, nr. 5837 of *LNCS*, 166–181. Springer.

Tappolet, J., Bernstein, A. 2009. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. *ESWC*, nr. 5554 of *LNCS*, 308–322. Springer.

Udrea, O.; Recupero, D. R.; Subrahmanian, V. S. 2006. Annotated RDF. *ESWC '06*, nr. 4011 of *LNCS*, 487–501. Springer.

[8] http://sites.google.com/site/annotatedrdf/