Annotated RDF*

Octavian Udrea, Diego Reforgiato Recupero, and V.S. Subrahmanian

University of Maryland, College Park MD 20742, USA
{udrea, diegoref, vs}@cs.umd.edu

Abstract. There are numerous extensions of RDF that support temporal reasoning, reasoning about pedigree, reasoning about uncertainty, and so on. In this paper, we present *Annotated RDF* (or aRDF for short) in which RDF triples are annotated by members of a partially ordered set (with bottom element) that can be selected in any way desired by the user. We present a formal declarative semantics (model theory) for annotated RDF and develop algorithms to check consistency of aRDF theories and to answer queries to aRDF theories. We show that annotated RDF captures versions of all the forms of reasoning mentioned above within a single unified framework. We develop a prototype aRDF implementation and show that our algorithms work very fast indeed - in fact, in just a matter of seconds for theories with over 100,000 nodes.

1 Introduction

Since the adoption of "Resource Description Framework" (RDF) as a web recommendation by the W3C, there has been growing interest in using RDF for knowledge representation [1, 2, 3, 4]. Extensions to RDF have included temporal extensions [5], fuzzy extensions [6, 7], provenance management methods [2], and others.

In this paper, we propose an extension of RDF called *Annotated RDF* (or aRDF for short) that builds upon *annotated logic* [8, 9] which has been subsequently used, extended and improved [10] for a wide range of knowledge representation tasks. In aRDF, you can start with any partially ordered set that you like as long as it has has a bottom element[1]. $\mathcal{A}$ could capture fuzzy or possibilistic values [2, 7] or timestamps [5] or - as we shall show - pedigree information or temporal-fuzzy information, and so on. We present a syntax for aRDF in Section 2 - in essence, an aRDF triple consists of an ordinary RDF triple together with an annotation (member of $\mathcal{A}$). We then present a declarative (model-theoretic) semantics for aRDF, together with notions of consistency and entailment in Section 3 — unlike ordinary RDF, an aRDF theory can be inconsistent and hence we provide a consistency check algorithm, together with

* Work supported in part by ARO grant DAAD190310202, AFOSR grant FA95500510298, the Joint Institute for Knowledge Discovery, and by a DARPA subcontract from the Univ. of California Berkeley.

[1] Suppose $(\mathcal{A}, \preceq)$ is a partially ordered set. $\perp \in \mathcal{A}$ is the "bottom element" of $\mathcal{A}$ iff $\perp \preceq x$ for all $x \in \mathcal{A}$.

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 487–501, 2006.

a result that whenever the partial order is a lattice, consistency is guaranteed. In Section 4, we present algorithms to answer three types of atomic queries, each with one unknown, together with an algorithm to answer conjunctive queries. We then present our prototype implementation and experiments in Section 5 — our experiments show that our framework is very efficient to implement in practice.

2 aRDF Syntax

We assume the existence of a partially ordered finite set $(\mathcal{A}, \preceq)$ where elements of $\mathcal{A}$ are called *annotations* and $\preceq$ is a partial ordering on $\mathcal{A}$. We further assume $\mathcal{A}$ has a bottom element. For example, we could have any of the following scenarios:

1. $\mathcal{A}_{fuzzy}$ may be the set of all real numbers in the closed interval $[0, 1]$ with the usual "less than or equals" ordering on it.
2. $\mathcal{A}_{time} = \mathbf{N}$ could be the set of all non-negative integers (denoting time points) with the usual "less than or equals" ordering on it.
3. $\mathcal{A}_{time-int} = \{[x, y] \mid x, y \in \mathbf{N}$ could be the set of all time intervals. The interval $[x, y]$ as usual denotes the set of all $t \in \mathbf{N}$ such that $x \leq t \leq y$. The inclusion ordering $\subseteq$ is a partial ordering on this set.
4. $\mathcal{A}_{pedigree}$ could be an enumerated set of sources with a partial ordering on them. If $s_1, s_2 \in \mathcal{A}_{pedigree}$, then we could think of $s_1 \preceq s_2$ to mean that s_2 has "better" pedigree than s_1.
5. $\mathcal{A}_{set-pedigree}$ could be the power set of $\mathcal{A}_{pedigree}$ with the Egli-Milner ordering which says that $S_1 \preceq S_2$ iff $(\forall s_1 \in S_1)(\exists s_2 \in S_2) s_1 \sqsubseteq s_2 \wedge (\forall s_2 \in S_2)(\exists s_2 \in S_1) s_1 \sqsubseteq s_2$. Note here that $\sqsubseteq$ is the ordering on $\mathcal{A}_{pedigree}$.
6. $\mathcal{A}_{fuztime}$ could be the set of all pairs (x, y) such that $x \in [0, 1]$ is a fuzzy value and y is a time point. The $\preceq$ ordering on $\mathcal{A}_{fuztime}$ can be defined as $(x, y) \preceq (x', y')$ iff $x \leq x'$ and $y \leq y'$.

These are just a few examples of partial orders. All the partial orders above except $\mathcal{A}_{pedigree}$ and $\mathcal{A}_{set-pedigree}$ are complete lattices[2]. Note that one can construct arbitrary combinations of partial orders by taking the Cartesian Product of two known partial orders and taking the pointwise ordering on the Cartesian Product as shown in the definition of $\mathcal{A}_{fuztime}$.

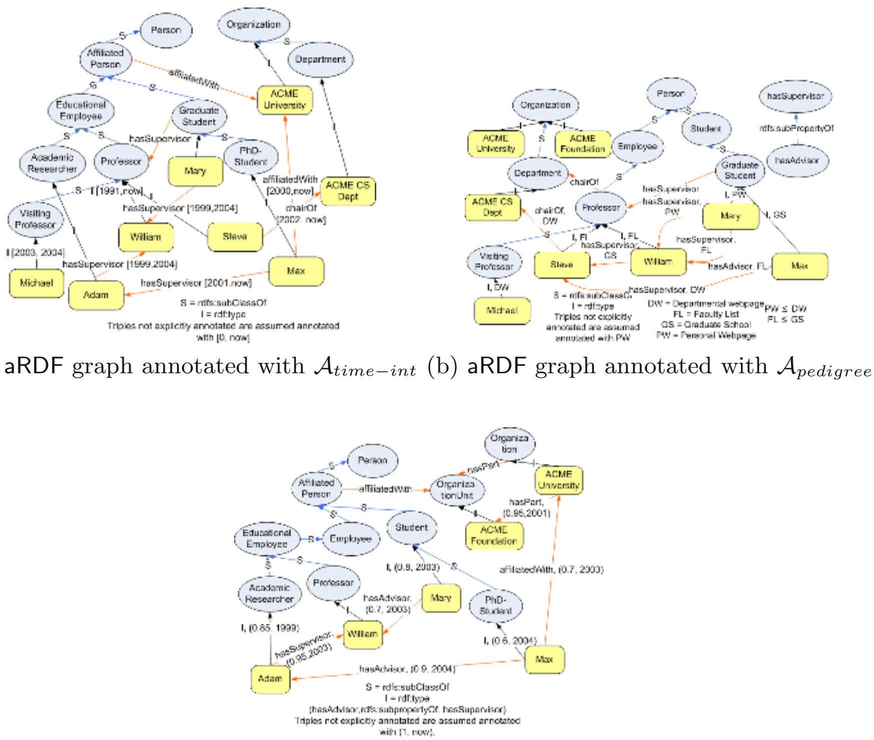
Suppose now that $(\mathcal{A}, \preceq)$ is an arbitrary but fixed partially ordered set. As in the case of RDF, we also assume the existence of some arbitrary but fixed set $\mathcal{R}$ of resource names, a set $\mathcal{P}$ of property names, and a set $dom(p)$ of values associated with any property name p.

An *annotated RDF-ontology* (aRDF-ontology for short)[3] is a finite set of triples $(r, p : a, v)$ where r is a *resource* name, p is a *property* name, $a \in \mathcal{A}$ and v is a

[2] A partially ordered set $(X, \leq)$ is a complete lattice iff (i) every subset of X has a unique greatest lower bound and (ii) every *directed* subset of X has a unique least upper bound. A set $Y \subseteq X$ is directed iff for all $y_1, y_2 \in Y$, there is an $x \in X$ such that $y_1 \leq x$ and $y_2 \leq x$.

[3] We will often abuse the term ontology to refer to both the intensional part (the schema) and the extensional part (the instance).

value (which could also be a resource name). In particular, this representation also supports RDF Schema triples such as[4]: (i) $(A, rdfs : subClassOf, B)$ indicates a subclass relationship between classes (which are also resources); (ii) $(X, rdf : type, C)$ indicates that a resource X is an instance of some class C; (iii) $(p, rdfs : subPropertyOf, q)$ denotes a sub-property relation between $p, q \in \mathcal{P}$[5]. We denote by $rdfs : subPropertyOf^*$ the reflexive, transitive closure of $rdfs : subPropertyOf$.[6] Once $\mathcal{R}, \mathcal{P}$ and $dom(\cdot)$ are fixed, we use the notation $Univ$ to denote the set of all triples (r, p, v) where $s \in \mathcal{R}, p \in \mathcal{P}$ and $v \in dom(p)$. *Throughout the rest of this paper, we will assume that* $\mathcal{R}, \mathcal{P}, \mathcal{A}, \preceq, dom(\cdot)$ *are all arbitrary, but fixed.*



(a) aRDF graph annotated with $\mathcal{A}_{time-int}$ (b) aRDF graph annotated with $\mathcal{A}_{pedigree}$

(c) aRDF graph annotated with $\mathcal{A}_{fuztime}$

Fig. 1. Three example aRDF ontology graphs

Definition 1. *(aRDF Ontology graph). Suppose O is an aRDF-ontology. An* aRDF ontology graph *for O is a labeled graph (V, E, λ) where*

---

[4] $rdfs : range$ and $rdfs : domain$ are also possible, as well as any other RDFS constructs. The paper focuses primarily on aRDF instances, therefore $rdfs : subPropertyOf$ schema constructs are particularly important.

[5] Note we did not require that $\mathcal{P} \cap \mathcal{R} = \emptyset$.

[6] We do not address reification and containers in RDF due to space constraints.

(1) $V = \mathcal{R} \cup \bigcup_{p \in \mathcal{P}} dom(p)$ *is the set of nodes.*
(2) $E = \{(r, r') \mid$ *there exists a property* p *such that* $(r, p : a, r') \in O\}$ *is the set of edges.*
(3) $\lambda(r, r') = \{p : a \mid (r, p : a, r') \in O\}$ *is the edge labeling function.*

It is easy to see that there is a one-to-one correspondence between aRDF-ontologies and aRDF-ontology graphs. Hence, we will often abuse notation and interchangeably talk about both aRDF ontologies and aRDF ontology graphs.

Example 1. Figure 1 shows three examples[7] of aRDF ontology graphs. Figure 1(a) is annotated with elements of $\mathcal{A}_{time-int}$. Therefore, the triple $(William, rdf : type : [1991, now], Professor)$ denotes the fact that William has been a Professor since 1991. Figure 1(b) uses $\mathcal{A}_{pedigree}$ for the annotation, with the partial order given in the figure. Here, the triple $(Steve, chairOf : DW, ACME\ CS\ Dept)$ denotes that the knowledge of Steve being the department chair was obtained from the department web page. Figure 1(c) is annotated with $\mathcal{A}_{fuztime}$ and contains both uncertainty and temporal information. For instance, the triple $(Adam, rdf : type : (0.85, 1999), AcademicResearcher)$ denotes that we are 85% certain that Adam was an academic researcher until 1999.

The rest of the paper will primarily focus on the semantics and query processing at the aRDF instance level; the problem of aRDF schema queries will be addressed in an extended version of this paper. We note that there are a number of ways in which aRDF theories can be represented in practice. One possible way is to use quadruples[8]; another possibility is the use of reification. Since aRDF semantics and query processing are the focus of this paper, we omit a lengthy discussion on representation issues.

As in the case of OWL, we differentiate between *transitive* and *non-transitive* properties. The RDFS semantics already specifies transitivity for rdfs:subClassOf and rdfs:subPropertyOf relations. The reader may view the specification of transitive properties as a poor man's inference capability for RDF instance data. We assume that all properties in $\mathcal{P}$ are marked transitive or non-transitive. For instance, in Figure 1(b) we consider *hasSupervisor* to be a transitive property[9].

Definition 2 (p-Path). *Let O be an aRDF ontology graph, p a transitive property in O and suppose $r, r' \in O$ are two nodes. There is a p-*path *between* r *and* r' *if there exist* $t_1 = (r, p_1 : a_1, r_1), \ldots, t_i = (r_{i-1}, p_i : a_i, r_i), \ldots, t_k = (r_{k-1}, p_k : a_k, r') \in O$ *such that* $\forall\ i \in [1, k]\ (p_i, rdfs : subPropertyOf^*, p)$. *We will denote a p-path Q by the set of triples $\{t_1, \ldots, t_k\}$ that form the path; we also say $A_Q = \{a_1, \ldots, a_k\}$ is the annotation of the p-path Q.*

Example 2. Consider the aRDF ontology graph shown in Figure 1(c) and suppose the *hasSupervisor* property is transitive. The triples $(Max, hasAdvisor :$

[7] In all examples, classes are represented with circular node and instances with rectangular nodes.
[8] A quadruple-based approach is currently discussed for representing contexts/data provenance in RDF — see http://www.w3.org/2001/12/attributions/.
[9] Although this is not generally the case, we assume this for the sake of the example.

$(0.9, 2004), Adam)$ and $(Adam, hasSupervisor : (0.95, 2003), William)$ form a *hasSupervisor*-path. Similarly, in Figure 1(b), assuming *hasSupervisor* and *hasAdvisor* are transitive properties, the triples $(Max, hasAdvisor : DW,$ $William)$ and $(William, hasSupervisor : GS, Steve)$ form a *hasSupervisor*-path, since $(hasAdvisor, rdfs : subPropertyOf, hasSupervisor)$.

3 aRDF Semantics

In this section, we provide a declarative semantics for aRDF ontologies and study consistency of such ontologies.

Definition 3. *An* aRDF-interpretation *I is a mapping from $Univ$ to $\mathcal{A}$.*

Definition 4. *An* aRDF-interpretation *I satisfies $(r, p : a, v)$ iff $a \preceq I(r, p, v)$. I satisfies an aRDF-ontology O iff:*

(S1) I satisfies every $(r, p : a, v) \in O$.
(S2) For all transitive properties $p \in \mathcal{P}$ and for all p-paths $Q = \{t_1, \ldots, t_k\}$ in O, where $t_i = (r_i, p_i : a_i, r_{i+1})$, and for all $a \in \mathcal{A}$ such that $a \preceq a_i$ for all $1 \leq i \leq k$, it is the case that $a \preceq I(r_1, p, r_{k+1})$.

O is consistent *iff there is at least one aRDF-interpretation that satisfies it. O* entails *$(r, p : a, v)$ iff every aRDF-interpretation that satisfies O also satisfies $(r, p : a, v)$.*

The definition of satisfaction and the complex definition of case (S2) above are best illustrated with an example.

Example 3. Let O be the aRDF ontology graph in Figure 1(c), where $\mathcal{A} = \mathcal{A}_{fuztime}$. Suppose the *hasSupervisor* property is transitive. Let $I_0(t) = (1, now)$ $\forall t \in Univ$. I_0 satisfies O and hence O is consistent. Furthermore, $O \models$ *(Mary, hasAdvisor: (0.7,2001), William)* because for any satisfying interpretation, *(0.7, 2001) $\preceq$ (0.7, 2003) $\preceq$ I(Mary, hasSupervisor, William)*.

The intuition behind item (S2) of Definition 4 is related to the notion of entailment. For instance, in Figure 1(c) — with *hasSupervisor* transitive —, from the triples $(Max, hasAdvisor : (0.9, 2004), Adam)$ and $(Adam, hasSupervisor :$ $(0.95, 2003), William)$, we can infer that with 90% probability, William was Max' supervisor until 2003, since $\forall\ (p, t) \in \mathcal{A}_{fuztime}$ s.t. $(p, t) \preceq (0.9, 2004)$ and $(p, t) \preceq (0.95, 2003)$ (i.e. $\forall\ (p, t) \preceq (0.9, 2003)$), $(p, t) \preceq$ *I(Max, hasSupervisor, William)*.

It is immediately clear from Definition 4 that unlike RDF ontologies which are always consistent, aRDF ontologies can be inconsistent. Consider the aRDF ontology graph in Figure 1(b) and assume the *hasSupervisor* property is transitive. We can identify the following sources of inconsistency:

1. The triples $(Mary, hasSupervisor : PW, William)$ and $(Mary, hasSuper-visor : FL, William)$[10] indicate that for any interpretation I, we cannot have

[10] The presence of such triples is reasonable since it indicates the same information was obtained from different sources for which we cannot compare the pedigree according to the partial order given.

that $PW \preceq I(Mary, hasSupervisor, William)$ and $FL \preceq$ *I(Mary, hasSupervisor, William)*, which contradicts item (S1) from Definition 4.
2. The presence of the different *hasSupervisor*-paths {*(Max, hasAdvisor:FL, William),(William, hasSupervisor:GS, Steve)*} and $\{(Max, hasSupervisor: DW, Steve)\}$ means that for any interpretation I, we cannot have that $FL \preceq I(Max, hasSupervisor, Steve)$ and $DW \preceq I(Max, hasSupervisor, Steve)$, thus contradicting item (S2) from Definition 4.

We now state a necessary and sufficient condition for checking consistency of an aRDF ontology.

Theorem 1. *Let O be an aRDF ontology. O is consistent iff:*

(C1) $\forall p \in \mathcal{P}$ *and* $\forall\ r, r' \in \mathcal{R}$ *such that* $\exists$ *distinct* $a_1, \dots a_k \in \mathcal{A}$ *and* $\forall i \in [1, k]\ \exists (r, p : a_i, r') \in O$, *then* $\exists\ a \in \mathcal{A}$ *s.t.* $\forall i \in [1, k]\ a_i \preceq a$ *AND*

(C2) $\forall p \in \mathcal{P}$ *transitive,* $\forall r, r' \in \mathcal{R}$, *let* $\{Q^1, \dots, Q^k\}$ *be the set of different p-paths between r and* r' *and let* $\{A_{Q^1}, \dots, A_{Q^k}\}$ *be the annotations for these p-paths. Let* $B_{Q^i} = \{a \in \mathcal{A} | a \preceq a'\ \forall a' \in A_{Q^i}\}$. *Then* $\exists\ a \in \mathcal{A}$ *s.t.* $\forall b \in \bigcup_{i \in [1,k]} B_{Q^i}, b \preceq a$[11].

The following result states that if we require $\mathcal{A}$ to be a partial order with a top element[12], then we are guaranteed consistency.

Corollary 1. *Let $\mathcal{A}$ be a partial order with a top element. Then any aRDF ontology O annotated w.r.t. $\mathcal{A}$ is consistent.*

The justification is immediate, since the interpretation that maps every triple in $Univ$ to the top element satisfies any aRDF ontology.

Theorem 1 provides an immediate algorithm for checking the consistency of aRDF ontologies. We present this algorithm in Figure 2.

Example 4. Let O the aRDF ontology graph in Figure 1(b). When we run our consistency check algorithm and execution reaches line 4 with *(r, p, r')=(Mary, hasSupervisor, William)*, $A = \{PW, FL\}$ from line 2. Since $\not\exists\ a \in \mathcal{A}$ s.t. $PW, FL \preceq a$, the algorithm will determine that the ontology is inconsistent.

Now consider the same aRDF ontology without the triple $(Mary, hasSuper-visor : PW, William)$. In this case, the algorithm will proceed to the loop starting on line 6. However, for the iteration for which $p = hasSupervisor$ on line 6 and $(r, r') = (Max, Steve)$ on line 9, the set P' will contain the two possible *hasSupervisor*-paths from Max to $Steve$ detailed in Example 3. Then on line 12, $A = \{\{DW\}, \{FL, GS\}\}$ and on line 13 $B = \{DW, FL\}$ and since $\nexists\ a \in \mathcal{A}$ s.t. $DW, FL \preceq a$, the algorithm will return $False$ on line 14.

The following result states the correctness of our consistency check algorithm.

Proposition 1 (Consistency check correctness). *The* $aRDFconsistency$ *on input* $(O, \mathcal{A}, \preceq)$ *returns* $True$ *iff O is consistent.*

[11] Note that (C2) implies (C1) when p is transitive, since paths of length 1 are possible.
[12] An element $\top \in \mathcal{A}$ is a "top" element if $x \preceq \top$ for all $x \in \mathcal{A}$.

Algorithm aRDFconsistency(O,$\mathcal{A}$, $\preceq$)
Input: aRDF ontology O and annotation ($\mathcal{A}$, $\preceq$).
Output: $True$ if O is consistent, $False$ otherwise.
Notation: For a property p we write $SP(p) = \{q \in \mathcal{P} | (q, rdfs : subPropertyOf^*, p)\}$. We denote by $O|_p$ the restriction of the aRDF graph O to triples labeled with properties in $SP(p)$. $N(O)$ denotes the set of nodes in the aRDF ontology graph O.

```
1. for (r, p, r′) ∈ {(r, p, r′)|∃ a ∈ 𝒜 s.t. (r, p : a, r′) ∈ O} do
2.    A ← {a ∈ 𝒜|(r, p : a, r′) ∈ O};
3.    if |A| > 1 then
4.       if ∄ a ∈ 𝒜 s.t. ∀a′ ∈ A, a′ ⪯ a return False;
5. end
6. for p ∈ 𝒫 transitive do
7.    O′ ← O|_p;
8.    P ← {paths Q ⊆ O′| ∄Q′ ⊆ O′ ∧ Q′ ⊃ Q};
9.    for (r, r′) ∈ N(O′) × N(O′) do
10.      P′ ← {Q ∈ P|r, r′ are the first and last node respectively in Q};
11.      if |P′| > 0 then
12.         A ← {A_Q|Q ∈ P′};
13.         B ← {b ∈ 𝒜|∃A_Q ∈ A s.t. ∀ a ∈ A_Q, b ⪯ a};
14.         if ∄ a ∈ 𝒜 s.t. ∀b ∈ B, b ⪯ a then return False;
15.      end
16.   end
17. end
18. return True;
```

Fig. 2. Consistency checking algorithm for aRDF ontologies

The consistency check algorithm runs in polynomial time as shown below.

Proposition 2 (Consistency check complexity). *Let O be an aRDF ontology graph and let $n = |N(O)|$, let $e = |O|$ and let $p = |\mathcal{P}|$. Let $(\mathcal{A}, \preceq)$ be a partial order and let $a = |\mathcal{A}|$[13]. Then aRDFconsistency$(O, \mathcal{A}, \preceq)$ is $\mathcal{O}(p \cdot (n^3 \cdot e + n \cdot a^2))$.*

The result follows from the loop on lines 6—17. For any transitive property, we first compute the set of all maximal paths in $O|_p$ (line 8). Since we have to keep the paths in memory (and not only their cost), this operation can be performed in at most $n^3 \cdot e$ steps in a modified version of Floyd's algorithm that records the paths explored. The loop on line 9 iterates through all the maximal paths found — there can be at most $2n$ of them. For each such path we compute the set A (line 12), which takes at most e steps, since any maximal path is of length less than or equal to e. The size of each A set is bounded by a and the number of maximal paths for the entire graph is at most $O(n)$, meaning line 13 will be run at most $\mathcal{O}(n \cdot a^2)$ times. Line 14 is run at most $\mathcal{O}(n \cdot a^2)$ times as well, since $|B|$ is bounded by a.

4 aRDF Query Processing

In this section, we consider aRDF-queries. We assume the existence of sets of variables ranging over resources, properties, values and $\mathcal{A}$. A term over one of these sets is either a member of that set or a variable ranging over that set. An *aRDF query* is a triple $(R, P : A, V)$ where R, P, A, V are all terms over

[13] We assume without loss of generality that $a < e$, since we can use at most one annotation for each edge.

resources, properties, annotations and values respectively. An aRDF query of the above form is atomic if at most one term in it is a variable.

Example 5. Consider the aRDF ontology graph in Figure 1(c). The following are aRDF atomic queries:

– What was the relationship between Max and William until 2002 with 80% probability? $(Max, ?p : (0.8, 2002), William)$.
– Who was Mary's supervisor until 2002 with 70% probability? *(Mary, hasSupervisor:(0.7,2002), ?v).*
– Who was affiliated with ACME University until 2002 with 65% probability? *(?r, affiliatedWith:(0.65,2002),ACME University).*

Definition 5 (Semi-unifiable aRDF triples). *Two aRDF triples* $(r, p : a, v)$, $(r', p' : a', v')$ *are* θ *semi-unifiable iff there exists a substitution* θ *such that* $r\theta = r'\theta$ *and* $p\theta = p'\theta$ *and* $v\theta = v'\theta$.

As usual, $r\theta$ denotes the application of θ to r.

Definition 6 (Query answer). *Let* O *be a consistent aRDF ontology and let* $q = (r_q, p_q : a_q, v_q)$ *be a query on* O. *Let* $A_O(q) = \{(r, p : a, v) \mid (r_q, p_q : a_q, v_q)$ *is semi-unifiable with* q *and* $O \models (r, p : a, v) \wedge ((a \text{ is a variable}) \vee (a_q \preceq a))\}$. *The* answer *to* q *is defined as* $Ans_O(q) = \{(r, p : a, v) \in A_O(q) \mid \nexists\ S \subseteq Ans_O(q) - \{(r, p : a, v)\}$ *s.t.* $S \models (r, p : a, v)\}$.

$A_O(q)$ consists of all ground (i.e. variable-free) instances of q that are entailed by O. However, $A_O(q)$ may contain redundant triples - for example, using our $time - int$ partial ordering, if $(r, p : [1, 100], v)$ is in $A_O(q)$, then there is no point including redundant triples such as $(r, p : [1, 10], v)$ in it. $Ans_O(q)$ eliminates all such redundant triples from $A_O(q)$.

Example 6. Consider the queries in Example 5. The answers are:

– $Ans_O(q) = \{(Max, hasSupervisor : (0.9, 2003), William)\}$. Note that the answer does not include for instance $(Max, hasSupervisor : (0.9, 2001), William)$ since the latter triple is already entailed by a triple in the answer.
– $Ans_O(q) = \{Mary, hasAdvisor : (0.7, 2003), William)\}$.
– $Ans_O(q) = \{Max, affiliatedWith : (0.7, 2003), ACME\ University)\}$.

The following result specifies a condition that must hold when O entails a ground aRDF triple.

Theorem 2. *Let* O *be a consistent aRDF ontology and let* $(r, p : a, v)$ *be an aRDF triple.* $O \models (r, p : a, v)$ *iff one of the following conditions holds:*

(E1) $\exists\ (r, p : a_1, v), \ldots, (r, p : a_k, v) \in O$ *and let* A *be the set of values* a' *such that* $a_i \preceq a'\ \forall i \in [1, k]$ *(*$|A| \geq 1$ *since* O *is consistent). Then* $\forall\ a' \in A$, $a \preceq a'$.

(E2) $\exists$ *p-paths* $Q^1, \ldots, Q^k$ *between* r *and* v. *Let* $B_{Q^i} = \{b \in \mathcal{A} | b \preceq a' \ \forall a' \in A_{Q^i}\}$. *Let* A *be the set of values* a' *such that* $\forall\ b \in \bigcup_{i \in [1,k]} B_{Q^i}, b \preceq a'$ *(*$|A| \geq 1$ *since* O *is consistent). Then* $\forall\ a' \in A$, $a \preceq a'$.

Given an ontology O, we can infer new triples from O using the following two operators, f_1, f_2:

1. $f_1(O) = \{(r, p : a, v) | \exists\ (r, p : a_1, v), (r, p' : a_2, v) \in O\ s.t.\ (p', rdfs : subPropertyOf^*, p) \wedge a$ is a minimal upper bound[14] of $a_1, a_2\}$.
2. $f_2(O) = \{(r, p : a, v) | \exists (r, p' : a_1, r'), (r', p'' : a_2, v) \in O\ s.t.\ (p', rdfs : subPropertyOf^*, p) \wedge (p'', rdfs : subPropertyOf^*, p) \wedge (\forall\ a' \in \mathcal{A}, (a' \preceq a_1 \wedge a' \preceq a_2) \Rightarrow (a' \preceq a)) \wedge (a\ minimal\ with\ these\ properties\ w.r.t.\ \preceq)\}$.

Let $\mu(O) = f_1(O) \cup f_2(O)$.

Proposition 3 (Closure of O**).** μ *is a monotonic operator, i.e.* $O_1 \subseteq O_2$ *implies* $\mu(O_1) \subseteq \mu(O_2)$. *Hence, by the Tarski-Knaster theorem, it has a least fixpoint denoted by lfp(O) called the* closure *of* O.

Example 7. Let O be the aRDF ontology in Figure 1(c). Then lfp(O) contains all triples in O and the triple *(Max, hasSupervisor: (0.9,2003), William)*.

The following result is a necessary and sufficient condition for entailment by an aRDF ontology.

Proposition 4. *Let* O *be an aRDF ontology.* $O \models (r, p : a, v)$ *iff* $(r, p : a, v) \in$ *lfp(O) or* $\exists (r', p' : a', v') \in$ *lfp(O) s.t.* $\{(r', p' : a', v')\} \models (r, p : a, v)$.

Proposition 5. *Let* O *be a consistent aRDF ontology and* q *a query on* O. *Then* $Ans_q(O) \subseteq$ *lfp(O).*

The above proposition gives us a very simple algorithm for answering queries.

1. Consider query $q = (r, p : a, v)$ on aRDF ontology O. Compute lfp(O).
2. $A \leftarrow \{(r', p' : a', v') \in \mathsf{lfp(O)} | (r', p' : a', v')\ semi-unifiable\ with\ q \wedge ((a\ is\ a\ variable) \vee (a \preceq a'))\}$.
3. Eliminate from A triples $(r, p : a, v)$ entailed by subsets of $A - \{(r, p : a, v)\}$.

However, we can do much better by avoiding the costly computation of lfp(O).

4.1 Answering Atomic Queries

Although the closure of an aRDF ontology gives a simple method of computing the answer to queries, its computation is potentially expensive. We show more efficient algorithms for answering atomic queries. The algorithm for queries of type $q = (r, p : a, ?v)$ is given in Figure 3; computing the answers to atomic queries of type $q = (?r, p : a, v)$ is very similar and omitted for reasons of space.

[14] a is an minimal upper bound of a_1, a_2 iff $a_1 \preceq a$ and $a_2 \preceq a$ and there is no other a' such that $a' \preceq a$ and $a_1, a_2 \preceq a'$.

Algorithm atomicAnswerV$(O, \mathcal{A}, \preceq, q)$
Input: Consistent aRDF ontology O, annotation $(\mathcal{A}, \preceq)$ and query $q = (r, p : a, ?v)$.
Output: $Ans_O(q)$.
Notation: For a property p we write $SP(p) = \{q \in \mathcal{P} | (q, rdfs : subPropertyOf^*, p)\}$. We denote by $O|_p$ the restriction of the aRDF graph O to triples labeled with properties in $SP(p)$.

1. $O \leftarrow O|_p$;
2. $Ans \leftarrow \emptyset$;
3. **if** p is non-transitive **then**
4.   **for** $(r, p', v') \in \{(r, p' : a', v') \in O\}$ **do**
5. $A \leftarrow \{a' \in \mathcal{A} | (r, p' : a', v') \in O\}$;
6. $B \leftarrow \{b \in \mathcal{A} | \forall a \in A, a \preceq b\}$;
7. $C \leftarrow \{c \in B | \nexists c' \in B, c' \neq c \ s.t. \ c' \preceq c\}$;
8. $Ans \leftarrow Ans \cup \{(r, p' : c, v') | c \in C \wedge a \preceq c\}$;
9.   **end**
10. **else if** p transitive **then**
11.   **for** all v' s.t. $\exists \ Q^1, \ldots, Q^k$ p-paths from r to v' **do**
12. $B \leftarrow \{b \in \mathcal{A} | \exists i \in [1, k] \ s.t. \ \forall \ a' \in \ A_{Q^i}, b \preceq a'\}$;
13. $C \leftarrow \{c \in \mathcal{A} | \forall b \in B, b \preceq c\}$;
14. $D \leftarrow \{d \in C | \nexists \ d' \in C, d' \neq d \ s.t. \ d' \preceq d\}$;
15. $Ans \leftarrow Ans \cup \{(r, p : d, v') | d \in D \wedge a \preceq d\}$;
16.   **end**
17. **end**
18. **return** Ans;

Fig. 3. Answering atomic aRDF queries $(r, p : a, ?v)$

Example 8. Consider the aRDF ontology graph in Figure 1(c) and the query $(Max, hasSupervisor : (0.8, 2002), ?v)$. Since *hasSupervisor* is transitive, the algorithm will go on the second branch, starting at line 10. The loop on line 11 iterates through all the values reachable through *hasSupervisor*-paths from Max, which are exactly {Adam, William}. Let us consider the second iteration, where $v' = William$. There is only one *hasSupervisor*-path between Max and $William$, containing triples $(Max, hasAdvisor : (0.9, 2004), Adam)$ and $(Adam, hasSupervisor : (0.95, 2003), William)$. Then $A_{Q^1} = \{(0.9, 2004), (0.95, 2003)\}$. Therefore B is exactly the set of pairs (p, t) s.t. $(p, t) \preceq (0.9, 2003)$. C will be the set of pairs (p, t) greater than $(0.9, 2003)$ and thus $D = \{(0.9, 2003)\}$. Therefore, the triple $(Max, hasSupervisor : (0.9, 2003), William)$ will be added to Ans.

The following theorem states that *atomicAnswerV* is correct.

Proposition 6. *$atomicAnswerV(O, \mathcal{A}, \preceq, q)$ returns $Ans_O(q)$.*

The following result says that *atomicAnswerV* runs in polynomial time.

Proposition 7. *Let O be an aRDF ontology graph and let n be the number of vertices in the ontology graph O, let $e = |O|$ and let $p = |\mathcal{P}|$. Let $(\mathcal{A}, \preceq)$ be a partial order and let $a = |\mathcal{A}|$. Then $atomicAnswerV(O, \mathcal{A}, \preceq, q)$ is $\mathcal{O}(n^2 \cdot e + n \cdot e \cdot a^2)$.*

The complexity result is given by the loop on lines 11—16. We start by determining all values reachable by p-paths from r and the corresponding paths, which can be done in $\mathcal{O}(n^2 \cdot e)$ since v is fixed. Since there are at most $\mathcal{O}(n)$ paths originating from r, each with at most $O(e)$ edges and the size of the annotation for each path is bounded by a, line 12 will be run at most $\mathcal{O}(n \cdot e \cdot a^2)$ times. Since the sizes of B, C, D are all bounded by a, the same result holds for lines 13—15.

Algorithm atomicAnswerP$(O, \mathcal{A}, \preceq, q)$
Input: Consistent aRDF ontology O, annotation $(\mathcal{A}, \preceq)$ and query $q = (r, ?p : a, v)$.
Output: $Ans_O(q)$.

$Ans \leftarrow \emptyset$;
for all p' such that $\exists\ Q^1, \ldots, Q^k$ p'-paths from r to v **do**
  $B \leftarrow \{b \in \mathcal{A} | \exists i \in [1, k]\ s.t.\ \forall\ a' \in\ A_{Q^i}, b \preceq a'\}$;
  $C \leftarrow \{c \in \mathcal{A} | \forall b \in B, b \preceq c\}$;
  $D \leftarrow \{d \in C |\ \nexists\ d' \in C, d' \neq d\ s.t.\ d' \preceq d\}$;
  $Ans \leftarrow Ans \cup \{(r, p' : d, v) | d \in D \wedge a \preceq d\}$;
end
return $\{(r', p' : a', v') \in Ans |\ \nexists\ S \subseteq Ans - \{(r', p' : a', v')\}\ s.t.\ S \models (r', p' : a', v')\}$;

Fig. 4. Answering atomic aRDF queries $(r, ?p : a, v)$

An even tighter complexity bound holds when the annotation is a complete lattice. In this case, after computing the set A on line 11, we can simply compute the least upper bound of the elements in A and thus obtain set C (on line 13). For complete lattices such as $\mathcal{A}_{time-int}$, this can be done in at most a linear number of steps in $|A|$. Thus, the overall complexity of the algorithm becomes $\mathcal{O}(n^2 \cdot e + n \cdot e \cdot a)$.

Algorithm *atomicAnswerP* given in Figure 4 computes the answer to atomic queries with an unknown property. The main difference from *atomicAnswerV* is that the graph we need to explore is the one containing all paths between r and v, instead of the one containing all p-paths starting at r. Depending on the shape of the aRDF ontology (e.g., breadth vs. depth), either search space may be larger, but the worst case complexity is identical. Algorithm *atomicAnswerA* given in Figure 5 computes the answer to atomic queries with unknown annotation. For *atomicAnswerA*, r, p, v are all known therefore the step in which we compute all paths (line 11) can be performed in at most $\mathcal{O}(n \cdot e)$ steps. Therefore, *atomicAnswerA* is $\mathcal{O}(n \cdot e \cdot a^2)$. Correctness results for both *atomicAnswerV* and *atomicAnswerA* similar to Proposition 6 are immediate.

Algorithm atomicAnswerA$(O, \mathcal{A}, \preceq, q)$
Input: Consistent aRDF ontology O, annotation $(\mathcal{A}, \preceq)$ and query $q = (r, p : ?a, v)$.
Output: $Ans_O(q)$.
Notation: For a property p we write $SP(p) = \{q \in \mathcal{P} | (q, rdfs : subPropertyOf^*, p)\}$. We denote by $O|_p$ the restriction of the aRDF graph O to triples labeled with properties in $SP(p)$.

1. $O \leftarrow O|_p$;
2. $Ans \leftarrow \emptyset$;
3. **if** p is non-transitive **then**
4.   **for** $(r, p', v) \in \{(r, p' : a', v) \in O | p' \in SP(p)\}$ **do**
5. $A \leftarrow \{a' \in \mathcal{A} | (r, p' : a', v) \in O\}$;
6. $B \leftarrow \{b \in \mathcal{A} | \forall a \in A, a \preceq b\}$;
7. $C \leftarrow \{c \in B |\ \nexists c' \in B, c' \neq c\ s.t.\ c' \preceq c\}$;
8. $Ans \leftarrow Ans \cup \{(r, p' : c, v) | c \in C\}$;
9.   **end**
10. **else if** p transitive **then**
11. $\{Q^1, \ldots, Q^k\} \leftarrow \{p\text{-paths from } r \text{ to } v\}$;
12. $B \leftarrow \{b \in \mathcal{A} | \exists i \in [1, k]\ s.t.\ \forall\ a' \in\ A_{Q^i}, b \preceq a'\}$;
13. $C \leftarrow \{c \in \mathcal{A} | \forall b \in B, b \preceq c\}$;
14. $D \leftarrow \{d \in C |\ \nexists\ d' \in C, d' \neq d\ s.t.\ d' \preceq d\}$;
15. $Ans \leftarrow Ans \cup \{(r, p : d, v) | d \in D\}$;
16. **end**
17. **return** Ans;

Fig. 5. Answering atomic aRDF queries $(r, p : ?a, v)$

The methods of computing query answers for atomic queries can be extended with minimal changes to the case of queries with multiple variables[15]; for reasons of space, we omit such algorithms.

4.2 Conjunctive Queries

Let O be a consistent aRDF ontology. We define conjunctive queries as a set $Q = \{q_1, \ldots, q_m\}$ of atomic queries, where $q_i = (r_i, p_i : a_i, v_i)$. The answer can be defined similarly to that of atomic queries as $Ans_O(Q) = \{S \subseteq O | \exists \theta\ s.t.\ \forall i \in [1, m], \exists (r, p : a, v) \in S\ s.t. ((r, p : a, v)\, \theta\ semi{-}unifiable\, with\ q_i) \wedge ((a_i\ variable) \vee (a_i \preceq a)) \wedge (\nexists S' \in Ans_O(Q)\ s.t.\ S' \models S)\}$. The algorithm for computing answers to conjunctive queries given in Figure 6 is based on the observation that a conjunctive query is apartially instantiated aRDF graph; thus, inexact graph matching algorithms [11] between the graph corresponding to Q and subgraphs of lfp(O) give potential answer sets.

Algorithm conjunctAnswer(O,$\mathcal{A}$, $\preceq$, Q)
Input: Consistent aRDF ontology O, annotation $(\mathcal{A}, \preceq)$ and query $Q = \{q_i = (r_i, p_i : a_i, v_i) | i \in [1, m]\}$.
Output: $Ans_O(q)$.
Notation: For a property p we write $SP(p) = \{q \in \mathcal{P} | (q, rdfs : subPropertyOf^*, p)\}$. We denote by $O|_p$ the restriction of the aRDF graph O to triples labeled with properties in $SP(p)$. $N(O)$ denotes the set of nodes in the aRDF ontology graph O.

if Q contains no variable property queries **then**
    $O \leftarrow O|_{\bigcup_i SP(p_i)}$;
$Ans \leftarrow \emptyset$;
do
    $O \leftarrow O'$;
    for all paths R in O on some property p between some r, r' **do**
        $B \leftarrow \{b \in \mathcal{A} | \forall\ a \in A_R, b \preceq a\}$;
        $C \leftarrow \{c \in \mathcal{A} | \forall\ b \in B, b \preceq c\}$;
        $D \leftarrow \{d \in C |\ \nexists\ d' \in C, d' \neq d, d' \preceq d\}$;
        $O' \leftarrow O \cup \{(r, p : d, r') | d \in D\}$;
    end
    for $(r, p, r') \in \{(r, p, r') | \exists\ a \neq a' \in \mathcal{A}\ s.t.\ (r, p : a, r'), (r, p : a', r') \in O\}$ **do**
        $A \leftarrow \{a \in \mathcal{A} | (r, p : a, r') \in O\}$;
        $B \leftarrow \{b \in \mathcal{A} | \forall a \in A, a \preceq b\}$;
        $C \leftarrow \{c \in B |\ \nexists c' \in B, c' \neq c\ s.t.\ c' \preceq c\}$;
        $O' \leftarrow O \cup \{(r, p : c, r') | c \in C \wedge a \preceq c\}$;
    end
while $O = O'$;
$G_Q \leftarrow$ the graph corresponding to Q;
for all matchings between G_Q and O **do**
    $ok \leftarrow true$;
    for $i \in [1, m]$ **do**
        $(r, p : a, v) \leftarrow$ the triple in O matched to q_i;
        if $\neg(a_i\ variable) \wedge \neg(a_i \preceq q)$ **then**
            $ok \leftarrow false$;
            break;
        end
    end
    if ok **then**
        $Ans \leftarrow Ans \cup \{$ set of triples matched to $G_Q\}$;
end
return Ans;

Fig. 6. Answering conjunctive aRDF queries

[15] However, the complexity of these algorithms remains polynomial.

Algorithm *conjunctAnswer* starts by computing the closure $\mathsf{lfp}(\mathsf{O})$ in the loop on lines 4—18. Elements corresponding to f_1 in Definition 3 are computed on lines 12—16, whereas elements corresponding to f_2 are computed on lines 6—11. After $\mathsf{lfp}(\mathsf{O})$ is computed, inexact graph matchings [11] are used to determine potential answers to the conjunctive query (line 20). Each triple in the potential answer is checked against the annotation (if constant) of the respective query (22—28). If all triples have "better" annotations than the corresponding query triples, the answer is stored (line 30). The complexity of *conjunctAnswer* is at worst case exponential since the computation of $\mathsf{lfp}(\mathsf{O})$ increases the size of the aRDF ontology polynomially and may be performed a number of times polynomial in the size of the ontology.

5 Experimental Results

Our experimental prototype of the aRDF query system was implemented in approximately 1100 lines of Java code; the experiments were performed on an Intel Pentium 4 Mobile processor machine at 2.30 GHz and 512MB DDR SDRAM, running Debian Linux 1.3.3.4-9. The experiments were run using synthetically generated aRDF datasets ranging from 10,000 to 100,000 aRDF triples, using an uniform distribution for the random generator. The following parameters were constant throughout the generation process: (i) $|\mathcal{P}| = 100$, (ii) 10 transitive properties, (iii) $|\mathcal{A}| = 20$, (iv) 10 subproperty relations.

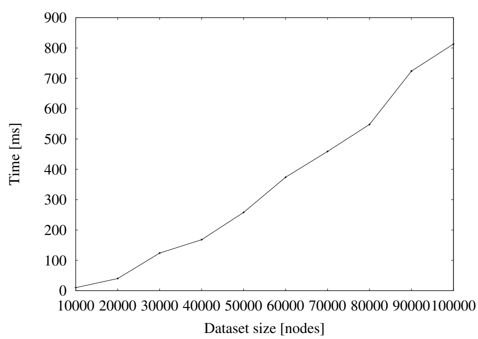
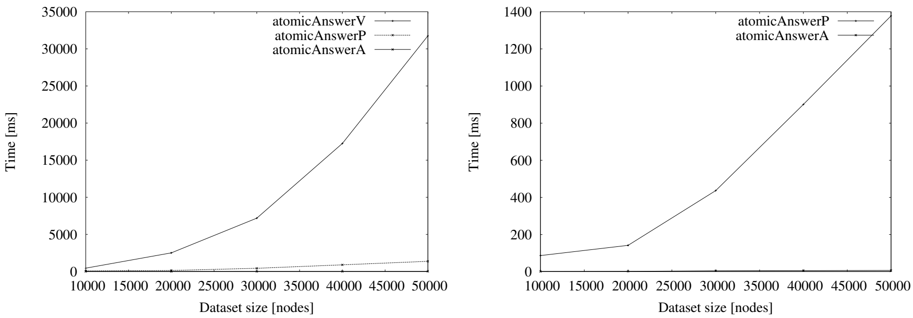


Fig. 7. aRDFconsistency running time

The first set of experiments shown in Figure 7 show the time needed for consistency checking. We see that aRDFconsistency takes under 1 second for graphs of 100,000 nodes. Figure 8(a) describes the query running time for the three algorithms detailed where queries were randomly generated. The main points that determine the behavior observed in Figure 8(a) and 8(b) are: (i) in line 11 of *answerV* we look for p-paths originating at a known r; (ii) line (2) of *answerP* we look for any transitive property paths between a known r and v; (iii) line (11) of *answerA* determines p-paths between a known r and v. It is



(a) Running time for atomic queries (b) Running time of $answerA$ and $answerP$

Fig. 8. Query running time

easy to see why (iii) is the fastest, since r, v, p are all known. We can also see that for the experimental setting described, (i) takes more time than (ii); this is due to the relatively small number of properties in the graphs[16].

6 Related Work

There has been considerable work on extending RDF with new features such as time intervals (statements saying something is true at all time points in an interval [5]), uncertainty [6, 7](though these are just one page position papers) and provenance [2] which describes a model for representing named RDF graphs, thus allowing statements about RDF graphs to be represented in RDF. [5] gives a model for temporal RDF, allowing triples to be specified as true for a finite time interval. [12] defines a model for representing multi-dimensional RDF, where information can be context dependent; for instance the title of a book may be represented in different languages. Our approach differs from all of the above: (i) we define a general framework for extending the RDF data model with annotations from an arbitrary partially ordered set; (ii) we give efficient algorithms for querying annotated RDF ontologies.

Our framework is based upon annotated logic [8, 9] — however, by examining RDF triples, we can provide far greater efficiency than annotated logic was able to provide. Moreover, annotated logic was unable to handle the kinds of queries shown where properties and the annotations desired were unknown.

To the best of our knowledge, this is the first paper that has attempted to provide a single framework - where by swapping a new partial order (with bottom) for another - we can get different types of reasoning capabilities in RDF. We have shown that annotated RDF is capable of supporting diverse forms of reasoning as well as combinations of reasoning (e.g. via `fuztime`), has a rich declarative semantics, and provides an efficient computational engine for application building.

[16] A phenomenon normally encountered in real-world RDF graphs, as we can see from most ontologies at `www.daml.org`.

References

1. Kahan, J., Koivunen, M.R.: Annotea: an open rdf infrastructure for shared web annotations. In: WWW '01: Proceedings of the 10th international conference on World Wide Web, New York, NY, USA, ACM Press (2001) 623–632
2. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW '05: Proceedings of the 14th international conference on World Wide Web, New York, NY, USA, ACM Press (2005) 613–622
3. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: Rql: a declarative query language for rdf. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM Press (2002) 592–603
4. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of semantic web databases. In: PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (2004) 95–106
5. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: Temporal rdf. In: ESWC. (2005) 93–107
6. D. Dubois, M., Prade, H.: Possibilistic uncertainty and fuzzy features in description logic: a preliminary discussion. In: Proc. Workshop on Fuzzy Logic and the Semantic Web (ed. E. Sanchez). (2005) 5–7
7. Straccia, U.: Towards a fuzzy description logic for the semantic web. In: Proc. Workshop on Fuzzy Logic and the Semantic Web (ed. E. Sanchez). (2005) 3–3
8. Kifer, M., Subrahmanian, V.S.: Theory of generalized annotated logic programming and its applications. J. Log. Program. **12** (1992) 335–367
9. Leach, S.M., Lu, J.J.: Query processing in annotated logic programming: Theory and implementation. J. Intell. Inf. Syst. **6** (1996) 33–58
10. Fitting, M.: Bilattices and the semantics of logic programming. J. Log. Program. **11** (1991) 91–116
11. Hlaoui, A., Wang, S.: A new algorithm for inexact graph matching. In: ICPR (4). (2002) 180–183
12. Gergatsoulis, M., Lilis, P.: Multidimensional rdf. In: Proc. 2005 Intl. Conf. on Ontologies, Databases, and Semantics (ODBASE). Volume 3761., Springer (2005) 1188–1205