Provenance, Trust, Explanations – and all that other Meta Knowledge

Renata Dividino, Simon Schenk, Sergej Sizov, Steffen Staab

Meta knowledge, i.e. knowledge about knowledge, is important in oder to judge the validity or appropriateness of the knowledge. We present a generic framework for managing and querying meta knowledge considering the most widespread paradigms for knowledge representation, i.e. RDF, Logic Programming, and Description Logics.

1 Introduction

One of the key benefits of the Semantic Web technology is the better support of decentralized, self-organizing knowledge exchange between users. When integrating knowledge from different sources or even from the full range of the Semantic Web, we are faced with highly varying quality of information. Hence, one of the major challenges is to investigate the value of information based on the trustworthiness of its sources, the time of validity, the certainty, or the vagueness asserted to explicitly specified or derived facts. In fact, this list of different types of 'knowledge about knowledge' (i.e. meta knowledge) is not even complete, one may be interested in the creator(s) of a fact, the document(s) in which it is stated — maybe implicitly, the web sites that certified it, and maybe many more dimensions. It becomes evident from such an enumeration that different types of meta knowledge should become accessible by a *generic framework* and not only by custom approaches that consider one particular dimension of meta knowledge at a time, possibly with incompatible assumptions.

It is the contribution of this paper to develop such a coherent framework by illustration with and reference to more specialized papers in the area of management of meta knowledge (in particular [1, 4, 10, 11]). To illustrate our framework let us first discuss a short example. Our sample application aims to suggest the local chair(s) for a multimedia congress in Koblenz, Rhineland-Palatinate. We may assume that the search for suitable candidates with relevant research profiles and appropriate location exploits collected knowledge from Semantic Web pages of multiple Computer Science departments.

Table 1 shows the instantiation for our sample scenario using a simplified abstract syntax for representing relevant facts and associated meta knowledge. We assume that all facts and axioms have been obtained from academic sites (University of Koblenz, FU Berlin, and Fraunhofer institutes) and some of them are also associated with last-modified timestamps (in the range between 2002 and 2008) which reflect the recency of knowledge. The collection presented in our example contains information about affiliation and research interests of academic scientists (e.g. the fact $\#_1$ represents the statement 'Research topic of Stefan Mueller is Computer Graphics'), as well as some definitions for the domain terminology (e.g. statement $\#_6$ defines a scientist to be a researcher working at the university) and rules for suggesting candidate chairs (e.g. statements $\#_8$ defines that 'Stefan Mueller is affiliated with only the University of Koblenz' and $\#_9$ postulates that the successful candidate should

work in Rhineland-Palatinate). The reader may note that the facts shown in Table 1 may require in practice the use of representation formalisms with different expressivity and complexity. While the facts $\#_1..\#_5$ can be expressed in RDF, the remainder requires more expressive frameworks like Logic Programming (for facts $\#_9, \#_{10}$) or OWL (for facts $\#_6..\#_8$).

An evident problem with the presented collection is that particular facts about affiliations and research interests (facts $\#_1..\#_5$) refer to *different* scientists, named likewise professor Stefan Mueller. The first one is a professor for Computer Graphics from the University of Koblenz, the second one is a professor for German Grammar of the University of Berlin. For this reason, some conclusions of our mix-up knowledge base may appear strange and curious. From the user's perspective, several sceptical questions may arise:

1. What are the research topic(s) of Stefan Mueller? Who said this (and when)?
2. May I trust the assessment that Stefan Mueller is a scientist?
3. What is the explanation that Stefan Mueller is the recommended candidate?

This paper points out the commonalities and differences between approaches that can be used to compute the solutions to these questions using meta knowledge. We consider three of the most widespread paradigms for knowledge representation, i.e. an algebraic approach, logic programming, and description logics. The differences result, first, from the different algebraic and logical mechanisms used for querying and inferencing upon a knowledge base. Second, the differences result from the algebraic or logic approaches used to formalize a theory for meta knowledge. The commonalities are found in the way in which we treat meta knowledge results computation. We represent meta knowledge by annotations of statements/axioms and combine such annotations in a versatile manner.

The design space of our treatment for meta knowledge is depicted in Figure 1. In the following, we instantiate in Section 2 the use of meta knowledge for algebraic querying methods (using our system Meta-K), in Section 3 for Logic Programming approaches (using the logic programming system Ontobroker [3]), and in Section 4 for OWL knowledge bases (using our system OWLMeta-K). The figure also indicates which combinations of knowledge and meta knowledge theories constitute straightforward extensions (using a '+') due to the decrease of the expressiveness power of the different theories for knowledge illustrated by the horizontal axis of Figure 1, while the logical analysis of meta knowledge for OWL will require further research.

ID	Relevant Facts	Meta Knowledge
$\#_1$	[stefanMueller researchTopic computerGraphics]	statedBy univKoblenz; statedBy fraunhofer
$\#_2$	[stefanMueller researchTopic eLearning]	statedBy univKoblenz; lastChange 2007
$\#_3$	[stefanMueller primaryAffiliation univKoblenz]	statedBy univKoblenz; lastChange 2002
$\#_4$	[rhineland-palatinate hasUniv univKoblenz]	statedBy univKoblenz
$\#_5$	[stefanMueller primaryAffiliation fuBerlin]	statedBy fuBerlin; lastChange 2007
$\#_6$	[scientist $\equiv \exists_{=1}$ primaryAffiliation.uni $\sqcap \exists$ researchTopic]	statedBy univKoblenz; lastChange 2008
$\#_7$	[uni $\equiv$ {fuBerlin, tuClausthal, univKarlsruhe, univKoblenz}]	statedBy univKoblenz
$\#_8$	[{stefanMueller} $\sqsubseteq \forall$ primaryAffiliation {univKoblenz}]	statedBy univKoblenz; lastChange 2008
$\#_9$	[candidateChair(X) :- primaryAffiliation(X,Y),orgOf(Y,rhineland-palatinate)]	explain "Person ?X is invited because he/she is with ?Y."
$\#_{10}$	[orgOf(S,T) :- hasUniv(T,S)]	

Table 1: Relevant facts that may have been obtained from different contexts
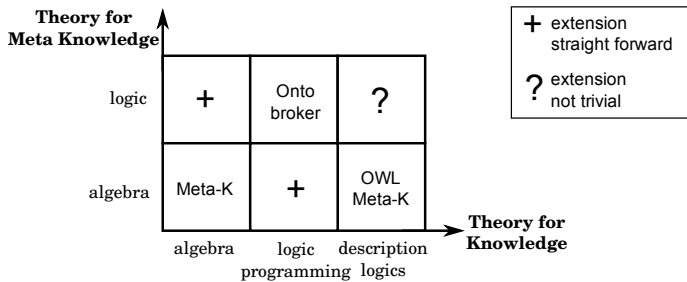

Figure 1: A Framework for Meta Knowledge Theories

2 Algebraic Meta Knowledge Interpretation for RDF and SPARQL

In this section we show a generic algebraic approach by re-using existing RDF modeling possibilities in order to represent meta knowledge. Furthermore, we integrate the meta knowledge management with SPARQL query processing and automatically extend the query output by associated meta knowledge.

2.1 Tracking How-Provenance in SPARQL

For querying RDF repositories with meta knowledge awareness, we exploit the capabilities of the SPARQL query language [9]. In doing so, we use in our framework **Meta-K** [12] two building blocks: algebraic semantics of SPARQL statements [8] (based on set-theoretic operations for sets of possible variable assignments), and the parallel computation of the *how-provenance* tracks (referring to the computation of *how* sources contribute to the existence of the data), represented by symbolic formulae (cf. [5]). For precisely tracking the provenance of query results and collecting associated meta knowledge, we exploit the notion of (internal) unique statement identifiers ($\#_1$ to $\#_5$, in our sample scenario). The formal grounding of the necessary RDF extension can be found in [12]. Basically, our approach extends the common algebraic SPARQL semantics [8], i.e. for annotated statements with symbolic statement identifiers.

The intermediate results of evaluating SPARQL expressions with meta knowledge can be represented in form of annotated relations. The schema of such relation includes all variables from the original query, and the instance contains all possible substi-

tutions of query variables by available RDF terms. In addition, we annotate each tuple of the instance by the individual how-provenance formula. This 'fingerprint' contains a combination of identifiers from statements that contributed to the tuple construction during query evaluation. The algebraic formalization and algorithms for evaluating SPARQL expressions are described in [12]. Basically, for algebraic operators used for SPARQL query evaluation [8], we define the use of corresponding algebraic operations $\wedge$, $\vee$, and $\neg$ for associated meta knowledge formulae constructed on statement identifiers.

We instantiate the presented approach by considering the SPARQL query shown in Example 2.1 in connection with statements $\#_1..\#_5$ from our sample collection of facts (Table 1). The example presents the query for the research topic(s) of Stefan Mueller who is affiliated with the University of Koblenz. It contains two triple patterns, $P_1 = $ *(?X primaryAffiliation univKoblenz)* and $P_2 = $ *(?X researchTopic ?Y)* which specify requirements to triples that can be used for instantiating variables $?X$ and $?Y$. In addition, the query explicitly restricts allowed instantiations of the variable $?X$ to *stefanMueller*, represented by the filter condition $?X = $ *stefanMueller*. According to the SPARQL semantics, all the specified requirements must hold in a conjunctive manner.

Example 2.1 *Sample SPARQL query*

```
SELECT ?X ?Y
WHERE{{
  GRAPH ?H {?X primaryAffiliation univKoblenz}
  GRAPH ?G {?X researchTopic ?Y}}
  {FILTER {?X = stefanMueller}}
```

The evaluation of this query in SPARQL with meta knowledge proceeds as follows. First, the patterns P_1 and P_2 are evaluated against statements $\#_1..\#_5$. All possible variable bindings for $?X$ and $?Y$ are represented in relational form and annotated by identifiers of the source triples. Subsequently, the evaluation of the conjunction between P_1 and P_2 (using the relational join operation ($result(P_1) \bowtie result(P_2)$)) and finally the application of the filter condition ($\sigma_{[?x=stefanMueller]}(result(P_1) \bowtie result(P_2))$) produces the annotated relation shown in Table 2.

The *how-provenance formulae* are constructed by combining annotations of tuples from source relations using the $\wedge$ operation. For instance, the formula of the first row ($\#_1 \wedge \#_3$) indicates that this assignment has been created using statements with identifiers $\#_1$ and $\#_3$.

$$\sigma_{[?X=stefanMueller]}\left(\begin{array}{|c|c|} \hline \texttt{?X} & \texttt{formulae} \\ \hline stefanMueller & \#_3 \\ \hline \end{array} \bowtie \begin{array}{|c|c|c|} \hline \texttt{?X} & \texttt{?Y} & \texttt{formulae} \\ \hline stefanMueller & computerGraphics & \#_1 \\ stefanMueller & eLearning & \#_2 \\ \hline \end{array} \right) =$$

?X	?Y	formulae
stefanMueller	computerGraphics	$\#_1 \wedge \#_3$
stefanMueller	eLearning	$\#_2 \wedge \#_3$

Table 2: Intermediate query processing results

2.2 Querying RDF with Meta Knowledge Awareness

The *how-provenance* clearly provides meaningful information about the origins of particular query results. However, from the end user perspective, it does not directly answer the practical questions about the result recency, its sources, or certainty. To link between how-provenance and relevant meta knowledge, custom interpretations of how-provenance formulas with respect to the available meta knowledge and its custom dimension are required. For each aspect of interest, we define the corresponding meta knowledge dimension as follows.

Definition 2.1 *Knowledge dimension for RDF. A knowledge dimension D is an algebraic structure $(\Omega_D, \vee_D, \wedge_D, \neg_D, \top, \bot)$, such that Ω_D is the value domain of D, $\vee_D, \wedge_D$ are custom user-defined binary functions on Ω_D, $\neg_D$ is a custom user-defined unary function on Ω_D, and $\top, \bot$ are two special elements.*

The definition of knowledge dimensions can be supplied by the administrator of the knowledge base, according to the intended semantics of particular meta knowledge aspects. Example 2.2 shows the definition of two meta knowledge dimensions from our sample scenario, namely last-modified timestamps (lastChange) and sources of knowledge (statedBy).

Example 2.2 *Possible interpretation for meta knowledge dimensions*

Meta dimension: lastChange (in milliseconds since 1970)

$\Omega_{lastChange} = [0, \infty)$
$I^f_{lastChange}(x_1 \wedge x_2) = \max(I^f_{lastChange}(x_1), I^f_{lastChange}(x_2))$
$I^f_{lastChange}(x_1 \vee x_2) = \min(I^f_{lastChange}(x_1), I^f_{lastChange}(x_2))$
$I^f_{lastChange}(\neg x_1) = 0$

Meta dimension: statedBy

$\Omega_{statedBy} = 2^D$, where D is the set of (legal) person URIs
$I^f_{statedBy}(x_1 \wedge x_2) = I^f_{statedBy}(x_1) \cup I^f_{statedBy}(x_2)$
$I^f_{statedBy}(x_1 \vee x_2) = I^f_{statedBy}(x_1) \cup I^f_{statedBy}(x_2)$
$I^f_{statedBy}(\neg x_1) = \{\}$

The introduced framework **Meta-K**[1] allows us to address the question 1) raised in Section 1: "What are the research topics of Stefan Mueller? Who said this (and when)?". Table 3 shows the evaluation results for our sample query, including variable assignments, the *how-provenance* formulae, and associated meta knowledge dimensions that were computed with respect to custom interpretation rules according to Example 2.2.

---

[1]**Meta-K** is available as an open source prototype at http://isweb.uni-koblenz.de/Research/MetaKnowledge

3 Logic Meta-Knowledge Interpretation — The Case of Explanations

In the previous section we have seen that we may use annotations of variable bindings of (partial) SPARQL queries in order to describe the how-provenance and then use the how-provenance in order to compute meta knowledge. When we go beyond querying and towards reasoning in logic programmes we may apply the same basic idea. The only difference to the previous section is that because of recursion we may encounter the same "query part", i.e. the same axiom, multiple times. Therefore, we must annotate nodes in the derivation tree of our query. Furthermore, we will show in this section that the basic idea of annotations cannot only be used by algebraic computations, but also by logic meta-programmes. In the example given in this section, we use it in order to compute explanations from the derivation tree.

3.1 Logic Programming

In current approaches for logic programming such as XSB, Ontobroker [3], and the DLV system [7], one specifies a logical theory using database facts and first-order Horn clauses with function symbols, possibly extended by means such as built-ins or negation. The knowledge specified in $\#_9$ and $\#_{10}$ constitute two such Horn clauses that can be applied to facts $\#_3$ and $\#_4$ in order to answer the query, who might possibly be a candidate for local chair of an event that is given to state employees of Rhineland-Palatinate, presented in Example 3.1.

Example 3.1 *Query example*

```
SELECT ?Z
WHERE { ?Z type candidateChair . }
```

In our logical programming approach, the query introduced in Example 3.1 could also be represented using the following representation: :- candidateChair(Z).

The answer would be derived by computing possible bindings (i.e. variable substitutions) for the variable Z, in this case the only possible binding would be (Z/stefanMueller). The actual computation involves (at least) one derivation tree for each possible variable binding. The shape of the derivation tree depends on the underlying semantics, e.g. the computation of a smallest fixpoint with three truth values for the well-founded semantics realized in Ontobroker [3] or in the Networked Graphs model [11]. For Horn clauses without negation, the example derivation tree is the same one would expect in Prolog (Figure 2), deriving the one possible answer for our running example.

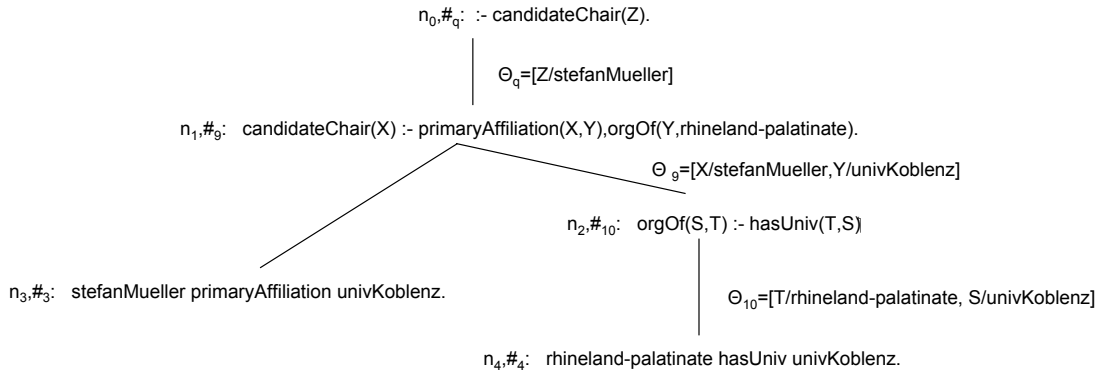| ?X	?Y	formulae	statedBy	lastChange
stefanMueller	computerGraphics	$\#_1 \wedge \#_3$	{univKoblenz, fraunhofer}	2002
stefanMueller	eLearning	$\#_2 \wedge \#_3$	{univKoblenz}	2007

Table 3: Query results with meta knowledge

Figure 2: Derivation tree for query: :- candidateChair(Z).

3.2 Computing Meta Knowledge for Logic Programmes

In [1], we have been able to demonstrate how to encode knowledge about anorganic chemistry in logic programs and how to derive facts and explanations from such knowledge theories and meta knowledge theories[2]. Following the same principled approach, we may use the annotations of derivation steps in Figure 2 in order to extend a meta knowledge theory with facts from a derivation tree and use the combination of the two in order to derive an explanation — or other kind of meta knowledge.

In order to encode the tree from Figure 2 we define its structure, as well as its individual bindings and axioms used in Table 4. Table 4 exhibits the similarity to the annotations of Table 2. The difference is that instead of a Boolean formula over statement identifiers, we have statement (i.e. axiom) identifiers in unison with an identifier for their use (i.e. the node identifier) and the variable bindings. Instead of some algebraic rules of how to combine annotations, we define a logic meta theory. In this example, the logic meta theory computes the explanations — however, it could also have been defined to compute dates of last changes or other knowledge dimensions.

Let us illustrate this step with our example. First, we re-state the meta-knowledge statement $\#_9$ in a logic programming syntax.

Example 3.2 *Meta knowledge for explanations*
explain($\#_9$, "Person ?X is invited because he/she is with ?Y.").

and we add the logic meta knowledge theory for our explanations. This theory allows for traversing a derivation tree and for collecting explanations of individual derivation steps. In this small example we do not use a lot of inferencing to come up with a good explanation, but it is not hard to see that such explanations could be tailored by corresponding axioms to consider

complex interdependencies between nodes in the tree or even the knowledge of the person asking the query:

Example 3.3 *Theory for explanations*
explainTree(N,S) :- explainNode(N,S), NOT andChildren(N,L).
explainTree(N,S) :- explainNode(N,S1), andChildren(N,L), explainSequence(L,S2), append(S1,S2,S).
explainSequence([], "").
explainSequence([H|T],S) :- explainNode(H,S1), explainSequence(H,S2), append(S1,S2,S).
explainNode(N,S) :- usesAxiom(N,A), explain(A,V), substituteBindingsInto(N,V).[4]

Finally, we can ask for explanations using the following query and receive the presented answer:

Example 3.4 *Explanation query and its respective answer*
:- explainTree(n_0,S).
[S/"Person stefanMueller is invited because he/she is affiliated with univKoblenz."]

4 Algebraic Meta-Knowledge Interpretation for OWL

In order to compute meta knowledge for an expressive logical formalism such as description logics, an algebraic approach as proposed in the Section 2 cannot be applied in a straight forward way. Such an application must fail, because the answer to a query in an expressive description logics in general cannot be computed algebraically. To overcome this problem, we decouple the process of finding answers for a query from the process of computing the meta knowledge for these answers.

---

[2]The **OntoNova** framework presented here has been implemented in the *Project Halo* and is available to download at http://www.projecthalo.com/downloads/improved/

[3]orChildren are defined analogously

[4]For sake of conciseness, we abstract here from the low level technical details of how to replace the variables found within the explanation string '?X' and '?Y' by the corresponding variable bindings found with the axioms used.

(a) Derivation tree from Figure 2 encoded as facts

andChildren(n_0, [n_1]).
andChildren(n_1, [n_2, n_3]).
andChildren(n_2, [n_4]).[3]

(b) Variable bindings for nodes in the derivation tree encoded as facts

usesAxiom(n_0, $\#_q$). hasBinding(n_0,Z,stefanMueller).
usesAxiom(n_1, $\#_9$). hasBinding(n_1,X,stefanMueller). hasBinding(n_1,Y,rhineland-Palatinate).
usesAxiom(n_2, $\#_{10}$). hasBinding(n_2,T,rhineland-palatinate). hasBinding(n_2,S,univKoblenz).
usesAxiom(n_3, $\#_3$).
usesAxiom(n_4, $\#_4$).

(c) Variable bindings for nodes in the derivation tree annotated with base explanations

Z	X	Y	T	S	Node	Axiom	explain
stefanMueller					n_0	$\#_q$	
	stefanMueller	rhineland-palatinate			n_1	$\#_9$	"Person ?X is invited because he/she is with ?Y".
			rhineland-palatinate	univKoblenz	n_2	$\#_{10}$	
					n_3	$\#_3$	
					n_4	$\#_4$	

Table 4: Derivation tree structure encoded as facts and its bindings

4.1 Tracking How-Provenance in OWL

To illustrate the difficulties arising when computing meta knowledge for OWL, consider the consequences of lines 3 and 5 to 7 in Table 1. Line 3 and 5 state affiliations for Stefan Mueller. According to line 6, a scientist must have exactly one primary affiliation, but according to line 7 the two affiliations of Stefan Mueller are disjoint. Hence, the answer to the query *scientist(stefanMueller)* must be false.

To reach this conclusion, however, no single distinguished model can be used. Even if we restrict interpretations to a fixed domain, there still can be multiple models. Moreover, while these models have the form of finite trees for less expressive DLs, for expressive DLs they may be infinite and have a forest shape. Hence, meta knowledge cannot be computed by simply traversing a derivation tree as we have seen in Section 2 for SPARQL or in Section 3 for the logic programming approach.

It is possible, however, to determine a finite set of axioms, which contribute to the answer; that is to find an explanation for the answer. This set must be finite, as it must be a part of the finite overall ontology. An explanation is a minimal set of axioms, which makes the concluded axiom true (or the theory inconsistent, respectively). Such an explanation is called a pinpoint. While there may be multiple ways to establish the truth or falsity of an axiom, a pinpoint describes exactly one such way. Hence, finding pinpoints for an unsatisfiable axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [6].

Pinpointing is the computation of all pinpoints for a given axiom and ontology. The truth of the axiom can then be computed using the *pinpointing formula* [2].

Definition 4.1 *Pinpointing Formula.*
Let A be an axiom, O an ontology and $P_1, ..., P_n$ with $P_i = \{A_{i,1}, ..., A_{i,m_i}\}$ the pinpoints of A wrt. O. Let lab be a function assigning a unique label to an axiom. Then $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} lab(A_{i,j})$ is a pinpointing formula of A wrt. O.

A pinpointing formula of an axiom A describes, which (com-bination of) axioms need to be true in order to make A true or inconsistent respectively. As a pinpointing formula describes how axioms can be combined to derive an answer to the query; and as meta knowledge is attached to the axioms, we have everything we need to compute meta knowledge for A.

Example 4.1 *Pinpointing Formula.*
The pinpointing formula for $\neg$ scientist(stefanMueller) *is* $\#_5 \wedge \#_6 \wedge \#_7 \wedge (\#_3 \vee \#_8)$.

4.2 Reasoning with Meta Knowledge in OWL

Meta knowledge can have multiple dimensions like uncertainty, last changed or a trust metric. We assume that these (and possible further) dimensions are independent of each other.

Definition 4.2 *Knowledge dimension for OWL.*
A knowledge dimension D is an algebraic structure $(\Omega_D, \vee_D, \wedge_D, \oplus_D)$, such that $(\Omega_D, \vee_D)$, $(\Omega_D, \wedge_D)$ and $(\Omega_D, \oplus_D)$ are complete semilattices. The minimum of $(\Omega_D, \oplus_D)$ is called $\bot_D$.

We use complete semilattices – i.e. algebraic structures, where the supremum of any two elements is well defined in order to ensure completeness of the language. The operator $\oplus_D$ represents the assumption that its operands both are assigned as meta knowledge to the same axiom, possibly resulting in an inconsistency, e.g. if we have two different *lastChange* dates. In contrast to inconsistencies in classical logics, such a situation does not render the whole theory useless. In contrast, $a \oplus_D b = \top$ means we cannot say anything useful about $a \oplus_D b$ in meta knowledge dimension D, but other meta knowledge dimensions or axioms may not be affected.

We associate ontology axioms with meta knowledge through OWL2 axiom annotations. Basically, an axiom annotation assigns an annotation object to an axiom e.g. "*primaryAffiliation(stefanMueller, univKoblenz)* was created by *UnivKoblenz* in 2002".

The meta knowledge of A is obtained from evaluating the corresponding pinpointing formula after replacing each state-

axiom	meta knowledge assignments	statedBy	lastChange
$\#_1$	statedBy univKoblenz; statedBy fraunhofer;	{univKoblenz,fraunhofer}	$\perp$
$\#_2$	statedBy univKoblenz; lastChange 2007	{univKoblenz}	2007
$\#_3$	statedBy univKoblenz; lastChange 2002	{univKoblenz}	2002
$\#_5$	statedBy fuBerlin; lastChange 2007	{fuBerlin}	2007
$\#_6$	statedBy uniKoblenz; lastChange 2008	{uniKoblenz}	2008
$\#_7$	statedBy uniKoblenz	{uniKoblenz}	$\perp$
$\#_8$	statedBy uniKoblenz; lastChange 2008	{uniKoblenz}	2008

Table 5: Meta knowledge for each axiom

query	pinpointing formula	statedBy	lastChange
$\neg scientist(stefanMueller)$	$\#_5 \wedge \#_6 \wedge \#_7 \wedge (\#_3 \vee \#_8)$	{univKoblenz, fuBerlin}	2008

Table 6: Pintpointing formula and meta knowledge for the evaluation of $\neg scientist(stefanMueller)$

query	pinpointing formula	statedBy	lastChange	trust
$scientist(stefanMueller)$	$\#_3 \wedge \#_6 \wedge \#_7 \wedge \#_8 \wedge (\#_1 \vee \#_2)$	{univKoblenz, fraunhofer}	2008	univKoblenz

Table 7: Evaluation of *scientist(stefanMueller)* after removing axiom $\#_5$

ment identifier with the corresponding meta knowledge assignments listed in Table 1.

Definition 4.3 *Meta Knowledge of an Axiom.*
Let meta *be a function mapping from an axiom to a meta knowledge assignment. The meta knowledge of an axiom A wrt. O is obtained by evaluating the formula obtained from A's pinpointing formula wrt. O by replacing each $lab(A_i)$ with the corresponding $meta(A_i)$.*

In our introductory scenario there may obviously be conflicting assignments of meta knowledge to axioms, as there could be concurrent modifications or repetitions of parts of the knowledge base by multiple users. Hence, we first need to merge these meta knowlege assignments.

Here, the formalization of meta knowledge analogous to logical bilattices comes into play: We redefine the *meta* function, such that it merges all meta knowledge assignments available for a statement using the $\oplus_D$ operator.

Definition 4.4 *meta.*
Let allmeta: axioms $\rightarrow 2^{MKAssignments}$ *be a function mapping from an axiom to all meta knowledge assignments to that axiom. Then $meta(A)$ is defined as $\bigoplus allmeta(A)$.*

This definition of *meta* not only allow for aggregating meta knowledge from multiple sources, but also to gracefully handle unknown meta knowledge, i.e. situations where a knowledges source does not provide a truth value for some meta knowledge dimension for a particular axiom. In such a case, $\perp_D$ is assumed as a default. In contrast to the SPARQL based instantiation of our framework[5], we omit the $\neg$ operator in our formalization. It is not needed, because description logics are monotonic and $\neg$ in Section 2 is used for default negation.

In oder to evaluate the meta knowledge formula for *scientist(stefanMueller)* we first determine the meta knowledge for each involved axiom in Table 5. Remember that the truth value

of *scientist(stefanMueller)* is false. Using the *statedBy* operators from Example 2.2 and maximum and minimum for $\vee$ and $\wedge$ of the *lastChange* dimension, we can compute the respective pintpointing formula for the query $\neg scientist(stefanMueller)$ as shown in Table 6. Now assume we know that Stefan Mueller indeed is a scientist. Hence, we need to remove one of $\{\#_5, \#_6, \#_7, (\#_3 \vee \#_8)\}$, ideally the least trusted one. We introduce a knowledge dimension $D = (\Omega_D, \vee_D, \wedge_D, \oplus_D)$ for trust with $\Omega_D = \{$fuBerlin, fraunhofer, univKoblenz, $\perp_D\}$, and a partial order over Ω_D, which reflects our preference for information about Stefan Mueller, which originates from Koblenz: $\{\perp_D <$ *fuBerlin* $<$ *univKoblenz*, $\perp_D <$ *fraunhofer* $<$ *univKoblenz* $\}$. $\wedge_D$ and $\vee_D$ are the minimum and maximum with respect to this order. We can now rank $\{\#_5, \#_6, \#_7, (\#_3 \vee \#_8)\}$ wrt. D and hence remove $\#_5$, which is assigned to "fuBerlin", and hence is the lowest trusted one. In this modified knowledge base, *scientist(stefanMueller)* evaluates to true as presented in Table 7.

5 Conclusion

In this paper we have demonstrated how to compute meta knowledge statements for answers to SPARQL queries on RDF knowledge bases, for inferred axioms in OWL and for answers derived by logical programs. The core idea of our framework is the computation of a structured justification, i.e. a tree corresponding to a query plan, a normal form corresponding to a union of pinpoints and a derivation tree for a set of variable bindings, respectively. We have shown that this basic idea carries very far and allows for computing different kinds of meta knowledge. The interested reader will also find in [12] reports on initial benign empirical results, as well as the discussion on the benefits for the user/developer. What we could not consider in this contribution is the intricate interaction between the querying or reasoning mechanism and the kind of meta knowledge computation. Depending on the regularities underlying justification structure and the meta knowledge framework, e.g. its algebraic properties, the results of computing meta knowledge may vary.

In general, it may vary depending on the concrete calculus and its ordering of derivation steps. However, for several combinations of query/reasoning mechanisms with algebraic meta knowledge frameworks previous work has shown the soundness and completeness of the meta knowledge framework (cf. [5, 4, 10]). Further work will be required to investigate the more complex interactions of logical meta knowledge theories with underlying inferences on knowledge theories, in particular.

References

[1] J. Angele, E. Mönch, H. Oppermann, S. Staab, and D.Wenke. Ontology-Based Query and Answering in Chemistry: OntoNova @ Project Halo. In *ISWC2003*, pages 913–928. Springer, 2003.

[2] F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. In *TABLEAUX '07*, pages 11–27. Springer, 2007.

[3] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semistructured information. In *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.

[4] R. Dividino, S. Sizov, and S. Staab. Managing RDF with Meta Knowledge Awareness. Technical report, University of Koblenz, 2009. `http://www.uni-koblenz.de/~dividino/Research/Publications/2009/metaknowledge.pdf`.

[5] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance Semirings. In *PODS '07*, pages 31–40. ACM, 2007.

[6] M. Horridge, B. Parsia, and U. Sattler. Laconic and Precise Justifications in OWL. In *ISWC2008*, pages 323–338. Springer, 2008.

[7] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.

[8] J. Perez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. In *ISWC2006*, pages 30–43. Springer, 2006.

[9] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2008. http://www.w3.org/TR/rdf-sparql-query/.

[10] S. Schenk, R. Dividino, and S. Staab. Reasoning With Meta Knowlege in Description Logics. Technical report, University of Koblenz, 2009. `http://www.uni-koblenz.de/~schenk/Research/Publications/2009/reasoningWithMetak.pdf`.

[11] S. Schenk and S. Staab. Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In *WWW '08*, pages 585–594. ACM, 2008.

[12] B. Schueler, S. Sizov, S. Staab, and D. Thanh Tran. Querying for Meta Knowledge. In *WWW'08*, pages 625–634. ACM, 2008.

Contact

Prof. Dr. Steffen Staab
Dr. Dr. Sergej Sizov
Simon Schenk
Renata Dividino
University of Koblenz-Landau
Dept. of Computer Science
ISWeb - Information Systems and Semantic Web
56070 Koblenz, Germany
Phone: +49 261 287 2712
Fax: +49 261 287 2721
Email: [staab,sizov,schenk,dividino]@uni-koblenz.de

Bild

Prof. Dr. Steffen Staab is professor for databases and information systems at the University of Koblenz-Landau, Germany, leading the research group on Information Systems and Semantic Web (ISWeb). **Dr. Dr. Sergej Sizov** joined the ISWeb group in 2006 as post-doctoral research fellow. His research interests lie in the fields of knowledge management, information retrieval and web mining. **Simon Schenk** and **Renata Dividino** work toward a PhD in the ISWeb group. Simon Schenk's research interests include trust and uncertainty on the Semantic Web, and distributed query evaluation. Renata Dividino's research interests focus on the investigation of the destination and provenance of data in distributed graphs in general and in the Semantic Web, in particular. Find more information about the ISWeb group at `http://isweb.uni-koblenz.de`