Reasoning With Provenance, Trust and all that other Meta Knowlege in OWL

Simon Schenk*, Renata Dividino* and Steffen Staab*
* ISWeb Research Group University of Koblenz-Landau
Email: sschenk, dividino, staab@uni-koblenz.de

***Abstract*—For many tasks, such as the integration of knowledge bases in the semantic web, one must not only handle the knowledge itself, but also characterizations of this knowledge, e.g.: (i) where did a knowledge item come from (i.e. provenance), (ii) what level of trust can be assigned to a knowledge item, or (iii) what degree of certainty is associated with it. We refer to all such kinds of characterizations as *meta knowledge*. Approaches for providing meta knowledge for query answers in relational databases and RDF repositories, based on algebraic operations, exist. As query answering in description logics in general does not boil down to algebraic evaluation of tree shaped query models, these formalizations do not easily carry over. In this paper we propose a formalization of meta knowledge, which is still algebraic, but allows for the computation of meta knowledge of inferred knowledge in description logics, including reasoning with conflicting and incomplete meta knowledge. We use pinpointing to come up with meta knowledge formulas for description logics, which then can be evaluated algebraically. We describe and evaluate our prototypical implementation.**

I. Introduction

When exploiting explicit/inferred knowledge in the semantic web, one must not only handle the knowledge itself, but also characterizations of this knowledge, e.g.: (i) where did a knowledge item come from (i.e. provenance), (ii) what level of trust can be assigned to a knowledge item, or (iii) what degree of certainty is associated with it. We refer to all such kinds of characterizations as *meta knowledge*. On the semantic web, meta knowledge needs to be computed along with each reasoning task.

Meta knowledge can come in various, complex dimensions. Many simplifications done today, such as assuming trust to be measured on a scale from 1 to 10, are not justified. In contrast, actual information sources, modification dates, etc. should be tracked to establish trust [1]. We propose a flexible mechanism for tracking meta knowledge, which meets these requirements.

Various approaches to this problem have been proposed. They can be grouped in to three clusters: First, we have extensions of logical formalisms, e.g. description logics, to deal with a particular kind of meta knowledge. Most prominent are extensions for reasoning with uncertainty, such as fuzzy and probabilistic [2] or possibilistic [3] description logics. Other proposals exist, which are tailored to specific meta knowledge such as trust [4]. Second, for systems allowing for algebraic query evaluation (such as relational databases and SPARQL engines), more flexible mechanisms such as [5] and [6] have been proposed, which allow for many kinds of meta knowledge, but are limited to lower expressiveness of the underlying logical formalism. Third, the expressive system proposed by [7] has a rather ad-hoc semantics, which is partially defined in constructors in queries and hence can differ in each query evaluation.

To come up with a flexible mechanism, which at the same time supports expressive logics and multiple kinds of meta knowledge, a suitable formalization of meta knowledge in a semantically precise manner is needed. Moreover, such a mechanism must be supported with a suitable operationalization. From the existing approaches it is clear, that integrating an expressive meta knowledge language with an expressive base knowledge representation language is a non-trivial task, mainly because of the different foundations, i.e. algebra vs. logics, of the meta knowledge and base languages.

Expressive descriptions of meta knowledge in less expressive languages (such as SPARQL based on RDF) have been founded on a tree-based algebraic formalization. Reasoning frameworks, however, frequently have non-tree-based derivations used for consistency checking and querying. In order to be able to reason with meta knowledge, which we formalize as algebraic structure, on top of expressive base languages, we propose a reasoning framework for meta knowledge based on *pinpointing*. Pinpointing summarizes explanations for axioms in a single boolean formula, which then can be evaluated using a meta knowledge algebra. We provide a blackbox algorithm for reasoning with meta knowledge, and describe our prototypical implementation. The algorithm uses an existing description logic reasoner for entailment checks. Hence, the supported expressivity is that of the underlying description logic.

As a motivation, we first explain a short use case, before laying foundations and defining the semantics of meta knowledge. Afterwards we briefly discuss the complexity and our prototypical implementation. We review the related work and conclude the paper.

II. Use Case

In a common scenario for collaborative ontology editing we have public, living ontologies, for which users can propose changes [8] and which are possibly interlinked through imports or views. Applications include large medical and biological ontologies such as SNOMED or the Gene Ontology. The example in [8] is based on a use case at the UN's Food and Agricultural Organization FAO. A change can be the addition, change, or removal of an axiom. Users have different levels

of expertise and hence their knowledge items are assigned different degrees of trustworthiness. Moreover, there may be conflicting changes or modifications, which make the ontology inconsistent. When answering queries and inferring knowledge in such systems, users need to know for example

- who contributed to axioms used to infer new knowledge,
- when they were last modified, and
- how trustworthy they are.

The derivation of meta knowledge can happen dynamically, in a completely open system comparable to today's wikis, where the change history is available for every user.

III. Foundations

As we use pinpointing as a vehicle for computing meta knowledge, we introduce pinpointing as a foundation for the rest of the paper and give some information of existing algorithms for finding pinpoints.

A. Pinpointing

The term pinpointing has been coined for the process of finding explanations for concluded axioms or for a discovered inconsistency. 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.

Definition 1: Pinpoint.
A pinpoint for a entailed axiom A wrt. an ontology O is a set of axioms $\{A_1, ..., A_n\}$ from O, such that $\{A_1, ..., A_n\} \models A$ and $\forall A_i \in \{A_1, ..., A_n\} : \{A_1, ...A_{i-1}, A_{i+1}, ..., A_n\} \not\models A$. Analogously, a pinpoint for a refuted axiom A wrt. an ontology O is a set of axioms $\{A_1, ..., A_n\}$ from O, such that $\{A, A_1, ..., A_n\}$ is inconsistent and $\forall A_i \in \{A_1, ..., A_n\} : \{A, A_1, ...A_{i-1}, A_{i+1}, ..., A_n\}$ is not.

Hence, finding pinpoints for a refuted axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [9].

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* [10].

Definition 2: 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 $\bigvee_{i=1}^{n} \bigwedge_{j=i}^{m_i} lab(A_{i,j})$ is a pinpointing formula of A wrt. O.

A pinpointing formula of an axiom A describes, which (combination of) axioms need to be true in order to make A true or inconsistent respectively.

B. Finding all Pinpoints

Algorithms for finding Pinpoints can be grouped into three groups:

a) Finding one pinpoint: Algorithms to find one pinpoint can either derive a pinpoint by tracking the reasoning process of a tableaux reasoner, or use an existing reasoner as a black box. In the latter case, a pinpoint is searched by subsequently growing (shrinking) a subontology until it starts (stops) entailing the axiom under question. Based on the so derived smaller ontology the process is refined, until a pinpoint has been found. The advantage of blackbox algorithms is that they can support any description logic, for which a reasoner is available [9]. Extending a tableaux reasoner on the other hand is complicated, but yields better performance, as a pinpoint can be generated in parallel to a usual subsumption check with low overhead [10].

b) Finding all Pinpoints using a Tableaux Reasoner: Baader and Peñaloza have shown that forest tableaux with equality blocking (and hence, reasoners for the web ontology language OWL) can be extended to find pinpointing formulas [10]. In this approach a tableaux reasoner is extended to find not only one, but all pinpoints. Special care needs to be taken in order to ensure termination of the tableaux algorithm. As an advantage, the overhead for pinpointing is lower compared to a blackbox algorithm. Moreover, this approach can derive a compact representation of the pinpointing formula, which might have worst-case exponential size in conjunctive normal form. To the best of our knowledge none of the standard reasoners for complex description logics has been extended in this direction yet.

c) Finding all Pinpoints using Blackbox Algorithms: The most performant black-box algorithms for finding all justifications first extract a relevant module from the overall ontology, ensuring that this module yields the same inferences with respect to the axiom on interest. Then, starting from a single pinpoint, which is computed using an algorithm discussed in paragraph III-B0a, Reiter's Hitting Set Tree algorithm [11] is used to compute all pinpoints by iteratively removing one axiom from the pinpoint at hand and growing it to a full pinpoint again [12], [13]. Using this kind of algorithm, a lot of subsumption checks in the underlying description logic are needed.

For both, tableaux based and black box algorithms, the worst case complexity of finding all pinpoints is rather high, as there can be exponentially many pinpoints for any given ontology. However, recent work has shown that in the average case, the number is significantly lower [10].

IV. Syntax of Meta Knowledge

Meta knowledge can be expressed as annotations on axioms. Annotations are of main importance for the management of ontologies as annotations may be used to support analysis during collaborative engineering.

We associate ontology axioms with meta knowledge through axiom annotations. Basically, an axiom annotation assigns an annotation object to an axiom e.g. "(brokenLimb subClass Limb) was created by Crow on 15.01.2008". A meta knowledge annotation consists of an annotation URI and a meta knowledge object specifying the value of the annotation. In our case, the meta knowledge object is a constant-value

TABLE I
EXAMPLE OF META KNOWLEDGE ASSOCIATED WITH AXIOMS.

ID	Relevant Facts	Meta Knowledge
$\#_1$	[limb1 Limb]	statedBy Crow; modified 14-01-2008
$\#_2$	[limb2 Limb]	statedBy Crow; modified 14-01-2008
$\#_3$	[limb1 isBroken true]	statedBy House; modified 15-01-2008;
$\#_4$	[limb2 isWrenched true]	statedBy House; modified 15-01-2008

representing who asserted/modified the axiom, when the axiom was last modified, or the uncertainty degree of the axiom, or a combination thereof. The grammar for meta knowledge annotations as an extension of OWL 2 annotations[1] is as follows:

OWLAxiomAnnotation := 'OWLAxiomAnnotation'
'('**OWLAxiom OWLAnnotation**$^+$')'
OWLAnnotation := **OWLConstantAnnotation**
OWLConstantAnnotation := **MetaKnowledgeAnnotation**
MetaKnowledgeAnnotation := 'MetaKnowledgeAnnotation'
'('**AnnotationURI MetaKnowledge**$^+$')'
MetaKnowledge := **CertaintyAnnotation** | **DateAnnotation** | **SourceAnnotation** | **AgentAnnotation**
CertaintyAnnotation := 'CertaintyAnnotation'
'('**AnnotationValue**')'
SourceAnnotation := 'SourceAnnotation' '('**AnnotationValue**')'
DateAnnotation := 'DateAnnotation' '('**AnnotationValue**')'
AgentAnnotation := 'AgentAnnotation' '('**AnnotationValue**')'

In our scenario we assume that we are looking for meta knowledge information about all limbs which are either broken or wrenched. Our ontology contains the axioms and meta knowledge annotations summarized in Table I.

An example of how meta knowledge is represented and associated with OWL axioms is presented below.

```
OWLAxiomAnnotation(ClassAssertion(limb1 Limb)
  MetaKnowledgeAnnotation(
    annot1 AgentAnnotation(Crow)))
OWLAxiomAnnotation(
  PropertyAssertion(limb1 isBroken true)
  MetaKnowledgeAnnotation(
    annot2 AgentAnnotation(House)))
```

Annotations, however, have no semantic meaning in OWL 2. All annotations are ignored by the reasoner, and they may not themselves be structured by further axioms. For this reason, as next step, we first define the semantics of meta knowledge, later we describe how meta knowledge can be combined with reasoning.

V. SEMANTICS OF META KNOWLEDGE

Meta knowledge can have multiple dimensions, e.g. uncertainty, a least recently modified date or a trust metric. For this paper, we assume that these (and possible further) dimensions are independent of each other.

[1]OWL 2 Web Ontology Language: Spec. and Func.-Style Syntax: http://www.w3.org/TR/2008/WD-owl2-syntax-20081202

Definition 3: Knowledge dimension. A knowledge dimension D is an algebraic structure $(B_D, \vee_D, \wedge_D)$, such that $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are complete semilattices.

B_D represents the values the meta knowledge can take, e.g. all valid dates for the least recently modified date or a set of knowledge sources for provenance. As $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are *complete* semilattices, they are, in fact, also lattices. Hence, there are minimal elements in the corresponding orders.

As an example, let I be the meta knowledge interpretation[2] that is a partial function mapping axioms into the allowed value range of a meta knowledge dimension, and A and B be axioms of an ontology such that $A \neq B$. Provenance, i.e. the set of knowledge sources a piece of knowledge is derived from, can be modeled as:

- $I(A \vee B) = I(A) \cup I(B)$
- $I(A \wedge B) = I(A) \cup I(B)$

The least recently modified date could be modeled as:

- $I(A \vee B) = min(I(A), I(B))$
- $I(A \wedge B) = max(I(A), I(B))$

Axioms can be assigned meta knowledge from any of the meta knowledge dimensions. Within a single assignment, the meta knowledge must be uniquely defined.

Definition 4: Meta Knowledge Assignment.
A meta knowledge assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of meta knowledge dimensions and corresponding truth values, such that $D_i = D_j \Rightarrow d_i = d_j$.

In our running example, the meta knowledge assignment for *PropertyAssertion(limb1 isBroken true)* is {*(agent, Crow), (date, 15.01.2008)*}

Without loss of generality we assume a fixed number of meta knowledge dimensions. As a default value for D_n in a meta knowledge assignment we choose $\bot_D$.

To allow for reasoning with meta knowledge, we need to formalize, how meta knowledge assignments are combined. *How provenance* [14] is a strategy, which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a boolean formula connecting the A_i. We call a logical formula expressing how provenance a *meta knowledge formula*. For example the following query finds all limbs, that are either broken or wrenched:

$$x : \text{Limb} \wedge (\langle x, \text{true}\rangle : \text{isBroken} \vee \langle x, \text{true}\rangle : \text{isWrenched}).$$

The results of this query and the corresponding meta knowledge formulas are:

$$\text{limb1} \mid \#_1 \wedge \#_3 \quad \text{and} \quad \text{limb2} \mid \#_2 \wedge \#_4$$

The operators for meta knowledge dimensions extend to meta knowledge assignments, allowing us to compute meta knowledge for entailed knowledge by evaluating the corresponding meta knowledge formula.

Definition 5: Operations on Meta Knowledge Assignments. Let A, B be axioms and $\text{meta}(A) = \{(D_1, x_1), ..., (D_n, x_n)\}$ and $\text{meta}(B) = \{(E_1, y_1), ..., (E_m, y_m)\}$ be meta knowledge

[2]The administrator defines the intended semantics of these properties in order to facilitate query processing with complex expressions and pattern combinations.

assignments. Let $\dim(A)$ be the set of meta knowledge dimensions of A. Then $\text{meta}(A) \vee \text{meta}(B) = \{(D, x \vee_D y) | (D, x) \in \text{meta}(A) \text{ and } (D, y) \in \text{meta}(B)\}$. $\wedge$ is defined analogously.

Having defined the operations on meta knowledge assignments, we can define formulas using these operations.

Definition 6: Meta Knowledge Formula.

Let A be an axiom of an ontology O, *lab* a function assigning a unique label to each A_i from O and *lab(O)* the set of all labels of axioms in O. A meta knowledge formula ϕ for a axiom A wrt. an ontology O is boolean formula over the set of labels $\{lab(A_1), ..., lab(A_n)\}$ of axioms $\{A_1, ..., A_n\}$ from O, such that for each valuation $V \subset lab(O)$, which makes ϕ true, the following holds: $lab^-(V) \models A$.

The meta knowledge of an axiom A within a meta knowledge dimension is obtained by evaluating the corresponding meta knowledge formula after replacing axiom labels with the corresponding meta knowledge in the dimension under consideration.

Definition 7: Meta Knowledge of an Axiom.

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

In our running example, if we model the agent dimension as where provenance, the meta knowledge of the query result for *limb1* is: *(agent, {Crow}) ∧ (agent, {House}) = (agent, {Crow} ∪ {House}) = (agent, {Crow, House}).*

In contrast to [5] we omit the $\neg$ operator in our formalization, as description logics are monotonic and $\neg$ in [5] allows for default negation. While axioms in the underlying description logic may contain negation, this negation is not visible on the level of meta knowledge.

VI. Extended Semantics for Conflicting Meta Knowledge

In the following we extend our model to support conflicting meta knowledge, which can arise from conflicting changes or meta knowledge assignments by multiple users in an axiom.

Definition 8: Extended knowledge dimension. A extended knowledge dimension D is an algebraic structure $(B_D, \vee_D, \wedge_D, \oplus_D)$, such that $(B_D, \vee_D)$, $(B_D, \wedge_D)$ and $(B_D, \oplus_D)$ are complete semilattices. The minimum of $(B_D, \oplus_D)$ is called $\bot_D$.

As an example, let I be the meta knowledge interpretation that is a partial function mapping axioms into the allowed value range of a meta knowledge dimension A be an axiom of an ontology, and I_1 and I_2 interpretations of multiple meta knowledge assertions to A. Provenance, i.e. the set of knowledge sources a piece of knowledge is derived from, can be modeled as:

- $I(A \oplus A) = I_1(A) \cup I_2(A)$

The least recently modified date could be modeled as

- $I(A \oplus A) = max(I_1(A), I_2(A))$

Consider the following example presented in Table II and assume that two users assert the same axiom at different times into the example ontology:

TABLE II
Extension of our scenario where we assume two users assert the same axiom at different times

ID	Relevant Facts	Meta Knowledge
$\#_1$	[limb1 Limb]	statedBy Crow; modified 14-01-2008
$\#_2$	[limb2 Limb]	statedBy Crow; modified 14-01-2008
$\#_3$	[limb1 isBroken true]	statedBy House; modified 15-01-2008;
$\#_4$	[limb2 isWrenched true]	statedBy House; modified 15-01-2008
.		
.		
.		
$\#_{10}$	[BrokemLimb subClassOf (isBroken true)]	statedBy Crow; modified 14-01-2008 statedBy House; modified 15-01-2008

In our running example, the meta knowledge assignment for axiom $\#_{10}$ is {*(agent, Crow), (date, 14.01.2008), (agent, House), (date, 15.01.2008)*}

In our running example, if we model the least recently modified date dimension, the meta knowledge of the axiom $\#_{10}$ is: *(date, {14.01.2008}) ⊕ (date, {15.01.2008}) = (date, max({14.01.2008}, {15.01.2008})) = (date, {15.01.2008}).*

Consider the extended semantics of meta knowledge, we need to describe a different way of finding a meta knowledge formula. We redefine the *meta* function of Definition 7, such that it computes $\bigoplus$ of all meta knowledge assignments available for a statement.

Definition 9: Meta Knowledge of an Axiom. Extended Definition.

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

This definition of *meta* not only allows to aggregate 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 example, we want to model the agent dimension as where provenance, the meta knowledge of the query result for: *ClassAssertion(BrokenLimb limb1).* The axiom is satisfiable, so the corresponding pinpointing formula is $\#_1 \wedge \#_3 \wedge \#_{10}$ = *(agent, {Crow}) ∧ (agent, House}) ∧ ((agent, {Crow}) ⊕ (agent, {House})) = (agent, {Crow, House}).*

VII. Computing Meta Knowledge using Pinpoints

In order to allow for an *algebraic* evaluation of meta knowledge dimensions, we need a *single* boolean formula. In meta knowledge mechanisms like [5], it is derived from queries in relational algebra. When reasoning with description logics, however, such a rather simple algebraic foundation of the basic language does not exist. Instead, multiple axioms may be needed to establish the truth or falsity of inferred knowledge. For this purpose, we have defined the *meta knowledge* formula in definitions 6 and 9.

As we can see above, definitions 2, 6 and 9 are quite similar. In fact, a pinpointing formula provides exactly what we need for a meta knowledge formula: All combinations of axioms, which can be used to establish the truth or falsity of inferred knowledge.

For this reason, when reasoning in a logic, where a pinpointing algorithm is known, we can compute a pinpointing formula and then derive meta knowledge as usual.

VIII. Complexity

The complexity of this rather naive approach for computing meta knowledge is equivalent to the computation of pinpointings. Due to the algebraic specification of meta knowledge the complexity of the meta knowledge formula is polynomial. If the meta knowledge formula is in conjunctive normal form, however, we might encounter an exponential blowup. Approaches for computing pinpointings like [10] which, rather than representing pinpoints formula in a conjunctive normal form, derive a compact representation of the pinpoints formula benefit the computation of meta knowledge since they avoid exponential blowup.

IX. Experiments

In this section, we present the evaluation results of our algorithm. The experiments were performed on a Windows XP SP3 System and 512MB maximal heap space was set. Sun's Java 1.5.0 Update 6 was used for Java-based tools.

Reasoning with Meta knowledge The framework for reasoning with meta knowledge is is available as a Java prototype and is available as an open source implementation at <http://isweb.uni-koblenz.de/Research/MetaKnowledge> together with example of ontologies extended with meta knowledge. The aggregation of meta knowledge is computed based on the model presented in Section VI and Section VII.

Reasoning with Pinpointing The framework for reasoning with pinpointing is implemented with the OWL API and the OWL-DL reasoner, Pellet[3]. Pellet provides the axiom pinpointing service for debugging ontologies that, for any arbitrary entailment derived by a reasoner from an OWL-DL knowledge base, returns the minimal set (explanations) of source axioms that cause an inconsistency and the relation between unsatisfiable concepts. The algorithm is black box based. In the following experiments we compare the processing time of our approach with reasoning with pinpointing approach.

Data Our sample data consists of 7 typical existing OWL ontologies used for debugging. This dataset has already been used for tests the computing time of laconic justifications in [15]. Table III shows the number of entailments that hold in them and provide the range of expresivity. Each ontology was classified in order to determine the unsatisfiable classes. This classes were selected as input (query) to compute the meta knowledge degree and pinpoints. For each query the time to compute all pinpoints and the meta knowledge degree was recorded.

[3]Pellet Reasoner: http://clarkparsia.com/pellet/

TABLE III
Ontologies uses in experiment. Table taken from [15]

ID	Ontology	Expressivity	Axioms	No. Entailments
1	Economy	$\mathcal{ALCH}(\mathcal{S})$	1625	51
2	People+Pets	$\mathcal{ALCHOIN}$	108	33
3	MiniTambis	$\mathcal{ALCN}$	173	66
4	Transport	$\mathcal{ALCH}$	1157	62
5	University	$\mathcal{SOIN}$	52	10
6	Chemical	$\mathcal{ALCHF}$	114	44
7	EarthRealm	$\mathcal{ALCHO}$	931	543

Evaluation Results Table IV displays the times for reasoning with meta knowledge and reasoning with pinpointing. For each ontology, we have computed all pinpoints for all unsatisfiable classes and reported the overall computing time. The experiments was done 10 times and the average time was considered. We can observe that the time for computing the meta knowledge degree takes longer than the computation of pinpointing (in average 4,9 ms longer). This is to be expected since the computation of meta knowledge degree is done once all justifications are already computed as we have shown in Section VIII. In all in all, the processing times presented in Table IV are still acceptable for interactive applications, and thus this approach can be used for solutions in real time.

TABLE IV
Times (in ms) to compute pinpointing vs. meta knowledge degree

ID	Ontology	Pinpointing	Meta Knowledge
1	Economy	347,63	348,24
2	People+Pets	328	329,12
3	MiniTambis	152,78	158,69
4	Transport	864,75	874,83
5	University	95,48	98,96
6	Chemical	3770,33	3781,17
7	EarthRealm	3030,06	3032,50

We expect optimizations to reduce the processing time to less than a second in the average case also for the more complex ontologies. As we are only interested in computing the meta knowledge, we can direct the pinpointing algorithm to only compute those pinpoints resulting in the highest meta knowledge values. The optimization will be reported in future work.

X. Related Work

Related work can be grouped into the following categories: (i) Extensions of description logics with a particular meta knowledge dimension, especially uncertainty. (ii) General meta knowledge for query answering with algebraic query languages. (iii) Extensions of description logics with general meta knowledge and (iv) meta knowledge for other logical formalisms.

ad (i) Several multi-valued extensions of description logic have been proposed: [2] propose fuzzy and probabilistic extensions of the DLs underlying the web ontology language OWL. [3] describe an extension towards a possibilistic logic. Another extension towards multi valued logic is presented by [4]. They target at trust and paraconsistency instead of uncertainty. OWL 2 is extended to reasoning over logical

bilattices. Bilattices which reflect the desired trust orders are then used for reasoning. [16] provide an extension to reasoning in OWL with paraconsistency.

All of these approaches have in common, that they modify the character of models in the underlying description logic, e.g. to fuzzy or possibilistic models. In our approach in contrast, we reason on a meta level: While the underlying model remains unchanged, we compute consequences of annotations on axioms. This meta level reasoning is not possible in the approaches proposed above. Unlike general meta knowledge, these approaches are more tailored to a specific need and hence reasoning is cheaper for some. Particularly for fuzzy, possibilistic and paraconsistent description logics, the complexity of the underlying logic carries over, while in our case additional complexity is introduced through pinpointing.

ad (ii) Meta knowledge to algebraic languages has been proposed by various authors, for example for the Semantic Web Query Language SPARQL [5] and for relational databases [6]. In [17] the authors have propose a framework for meta knowledge management with support for querying and updating RDF/S graphs that takes into account both RDF named graphs and RDFS inference. While the actual meta knowledge formalisms are comparable to ours, the underlying languages are of lower expressivity, typically Datalog. Meta knowledge formulas in these language can directly be derived from the tree shaped representation of a query, which is not possible in description logics.

ad (iii) [7] propose a meta knowledge extension of OWL, which is also based on annotation properties. Even though meta knowlege can be expressed in ways comparable to ours, it has a rather ad-hoc semantics, which may differ from query to query. In our approach, meta knowledge and classical reasoning take place in parallel. Hence, we can answer queries such as "Give me all results with a confidence degree of $\geq x$". In contrast, reasoning on the ontology and meta level in [7] is separated. As a result, queries such as the following can be answered: "Give me all results, which are based on axioms with a confidence degree of $\geq x$". Although this difference might seem quite subtle, depending on the meta knowledge dimension, e.g. probabilistic confidence, these queries may have very different results.

ad (iv) [18] propose an extension of Datalog with weights, which are based on c-semirings and can be redefined to reflect various notions of trust and uncertainty. Our meta knowledge dimensions are similar to c-semirings, but additionally allow to handle conflicting meta knowledge using a third operator. As c-semirings have been investigated in great detail and have some desirable properties[4], a modification of our work towards similar algebraic structures might introduce additional interesting properties of meta knowledge.

XI. Conclusion

We have introduced a formalization of meta knowledge that allows to handle conflicting and incomplete meta knowledge on the Semantic Web. Meta knowledge per se cannot easily be built into a logical formalism such as description logics. Hence, we have provided an operationalization based on pinpointing, in order to derive a meta knowledge formula, which can easily be evaluated. Extensions of the approach beyond description logics are possible, based on pinpointing. Currently, we are working on the optimization of the algorithms for computing meta knowledge. The optimization are possible based on the observation, that we no longer need to compute all pinpointing formulas in oder to determine the meta knowledge but only computing a relevant subset of all pinpoints.

[4]Such as the fact that the cartesian product of two c-semirings again is a c-semiring.

XII. Acknowledgements

This research was supported by the European Commission under contract IST-FP6-026978, X-Media and contract IST-2006-027595, Lifecycle Support for Networked Ontologies - NeOn. The expressed content is the view of the authors but not necessarily the view of the X-Media and NeOn projects.

References

[1] Harry Halpin: Provenance: The Missing Component of the Semantic Web, CEUR Workshop Proceedings, online CEUR-WS.org/Vol-447/paper1.pdf

[2] Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. Journal of Web Semantics **6**(4) (2008) 291–308

[3] Qi, G., Pan, J.Z., Ji, Q.: Extending Description Logics with Uncertainty Reasoning in Possibilistic Logic. In: ECSQARU '07, Springer (2007) 828–839

[4] Schenk, S.: On the Semantics of Trust and Caching in the Semantic Web. In: ISWC2008. Volume 5313 of LNCS., Springer (2008) 533–549

[5] Schueler, B., Sizov, S., Tran, D.T.: Querying for Meta Knowledge . In: WWW2008, ACM (2008) 625–634

[6] Buneman, P., Khanna, S., Tan, W.C.: Why and Where: A Characterization of Data Provenance. In: ICDT. Volume 1973 of LNCS. (2001) 316–330

[7] Tran, D.T., Haase, P., Motik, B., Cuenca-Grau, B., Horrocks, I.: Metalevel Information in Ontology-Based Applications. In: AAAI'08. (2008) 1237–1242

[8] Palma, R., Haase, P., Corcho, Ó., Gómez-Pérez, A., Ji, Q.: An Editorial Workflow Approach For Collaborative Ontology Development. In: ASWC2008. Volume 5367 of LNCS. (2008) 227–241

[9] Kalyanpur, A., Parsia, B., Cuenca-Grau, B., Sirin, E.: Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL. Technical report (2006)

[10] Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. In: TABLEAUX '07: Proceedings of the 16th international conference on Automated Reasoning with Analytic Tableaux and Related Methods, Berlin, Heidelberg, Springer-Verlag (2007) 11–27

[11] Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1) (1987) 57–95

[12] Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of owl dl entailments. In: ISWC/ASWC. (2007) 267–280

[13] Ji, Q., Qi, G., , Haase, P.: A relevance-based algorithm for finding justifications of DL entailments. Technical report, University of Karlsruhe (2008)

[14] Green, T.J., Karvounarakis, G., Tannen, V.: Provenance Semirings. In: PODS. (2007) 31–40

[15] Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in owl. In: ISWC '08: Proceedings of the 7th International Conference on The Semantic Web, Berlin, Heidelberg, Springer-Verlag (2008) 323–338

[16] Ma, Y., Hitzler, P., Lin, Z.: Algorithms for Paraconsistent Reasoning with OWL. In: ESWC2007, Springer (2008) 399–413

[17] Pediaditis, P., Flouris, G., Fundulaki, I., Christophides, V.: On explicit provenance management in rdf/s graphs. In: TAPP'09: First workshop on on Theory and practice of provenance, Berkeley, CA, USA, USENIX Association (2009) 1–10

[18] Bistarelli, S., Martinelli, F., Santini, F.: A Semantic Foundation for Trust Management Languages with Weights: An Application to the RT Family. In: ATC '08, Springer (2008) 481–495