Querying for Meta Knowledge

Bernhard Schueler
Sergej Sizov
Steffen Staab

Nr. 8/2008

Arbeitsberichte aus dem
Fachbereich Informatik

The "Arbeitsberichte aus dem Fachbereich Informatik" comprise preliminary results which will usually be revised for subsequent publication. Critical comments are appreciated by the authors. All rights reserved. No part of this report may be reproduced by any means or translated.

**Arbeitsberichte des Fachbereichs Informatik**

**ISSN (Print):** 1864-0346  
**ISSN (Online):** 1864-0850

**Herausgeber / Edited by:**

Der Dekan:  
Prof. Dr. Zöbel

Die Professoren des Fachbereichs:  
Prof. Dr. Bátori, Prof. Dr. Beckert, Prof. Dr. Burkhardt, Prof. Dr. Diller, Prof. Dr. Ebert, Prof. Dr. Furbach, Prof. Dr. Grimm, Prof. Dr. Hampe, Prof. Dr. Harbusch, Jun.-Prof. Dr. Hass, Prof. Dr. Krause, Prof. Dr. Lämmel, Prof. Dr. Lautenbach, Prof. Dr. Müller, Prof. Dr. Oppermann, Prof. Dr. Paulus, Prof. Dr. Priese, Prof. Dr. Rosendahl, Prof. Dr. Schubert, Prof. Dr. Staab, Prof. Dr. Steigner, Prof. Dr. Troitzsch, Prof. Dr. von Kortzfleisch, Prof. Dr. Walsh, Prof. Dr. Wimmer, Prof. Dr. Zöbel

**Kontaktdaten der Verfasser**

Bernhard Schueler, Sergej Sizov, Steffen Staab  
Institut für Informatik  
Fachbereich Informatik  
Universität Koblenz-Landau  
Universitätsstraße 1  
D-56070 Koblenz  
EMail: bernie@uni-koblenz.de, sizov@uni-koblenz.de, staab@uni-koblenz.de
ABSTRACT
The Semantic Web is based on accessing and reusing RDF data from many different sources, which one may assign different levels of authority and credibility. Existing Semantic Web query languages, like SPARQL, have targeted the retrieval, combination and reuse of facts, but have so far ignored all aspects of meta knowledge, such as origins, authorship, recency or certainty of data, to name but a few.

In this paper, we present an original, generic, formalized and implemented approach for managing many dimensions of meta knowledge, like source, authorship, certainty and others. The approach re-uses existing RDF modeling possibilities in order to represent meta knowledge. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request meta knowledge without modifying the query proper. Thus, our approach achieves highly flexible and automatically coordinated querying for data and meta knowledge, while completely separating the two areas of concern.

1. INTRODUCTION
Integrating and re-using Semantic Web data becomes more and more fruitful and worthwhile in order to answer questions and deliver results. Typically, engines like Swoogle provide points of access for RDF data, crawlers may fetch relevant RDF data, and query languages like SPARQL with their corresponding query engines allow for selecting and re-using data in the appropriate format. With the arrival of more and more data in the Semantic Web and more sophisticated processing through query and reasoning engines, one now, however, encounters challenging questions linked to meta knowledge about the data like:

- Where is this data from?
- Who provided the data?
- When was this data provided?
- Was the provider certain about the truth of this data?
- Was the data believed by others, too?

For instance, when querying the Semantic Web with the help of SPARQL for the affiliation of a person named of “James Hendler”, one finds (at least) two answers, i.e. “University of Maryland” and “Rensselaer Polytechnic Institute”. Without further indication as to where, by whom, when, etc. such information was given, it is impossible to decide which of the two affiliations is still valid.

The problem might be remedied in several ways. First, an ideosyncratic solution by the search engine, such as returning the corresponding RDF files or links to sources of knowledge extraction (say http://www.cs.umd.edu/survey.pdf and http://www.rpi.edu/report.doc), might help in this special case. However, an ideosyncratic solution may not be appropriate in a second case in which the ‘when’ was more relevant than the ‘where’ or in a third case where such a piece of information had to be aggregated from several resources. Second, the person or system requesting the meta knowledge might manually extend the SPARQL query formalizing the request for the affiliation in order to return the where, the who and the when. Such a modification will, however, be very tedious, as it will include a number of additional optional statements, and expressing it manually will be error prone. Also, it will not help in delivering meta knowledge that arises from joining several statements, e.g. meta knowledge about uncertainty that was based on several meta knowledge statements with different values of uncertainty.

Therefore, querying Semantic Web data requires a principled, generic approach to the treatment of meta knowledge that is able to adapt to many dimensions of meta knowledge and that is open to accommodate to new dimensions when the need arises. Such a principled, original framework is given in this paper. We start to explain our approach with a discussion of important design choices in section 2. We model meta knowledge in existing RDF structures by embedding a slightly more expressive language, which we call RDF\^\textsuperscript{+}, into RDF. We define the abstract syntax of RDF\^\textsuperscript{+}, its semantics and its embedding in RDF in Section 4. In Section 5, we extend the SPARQL syntax and semantics to work on data and meta knowledge of RDF\^\textsuperscript{+}. The extension allows the user to extend a given conventional SPARQL query by a keyword for meta knowledge triggering the construction of meta knowledge by the query processor. Section 6 summarizes the overall use and processing of SPARQL queries with meta knowledge. Section 7 reports on initial graceful results for meta knowledge processing from a theoretic point of view and Section 8 provides pointers to the prototype implementation of the system.
2. SCENARIO
In our sample application scenario, we assume that the user utilizes knowledge which has been initially extracted from Web pages of Computer Science departments and stored in form of RDF triples in his personal “active space” [17], backed by a local RDF repository. Example 2.1 shows the relevant facts that may have been obtained from departments of different universities. For better readability, we use for our examples in this paper the RDF triple language TriG [1] with Named Graphs [2] in a simplified form that abstracts from default namespaces.

**Example 2.1. Extracted Knowledge and SPARQL query**

\[
\text{G1} \{ \text{JamesHendler researchTopic SemanticWeb} . \\
\text{JamesHendler affiliatedWith RensselaerPI} \}
\]

\[
\text{G2} \{ \text{JamesHendler researchTopic Robotics} . \\
\text{JamesHendler affiliatedWith UnivMaryland} . \\
\text{RudiStuder researchTopic SemanticWeb} . \\
\text{RudiStuder affiliatedWith UnivKarlsruhe} \}
\]

The extracted knowledge comes from different sources, at different timepoints, and with different degrees of extraction confidence. This information is also captured and stored into the same RDF repository as shown in example 2.2, using the notion of Named RDF Graphs [2, 5].

**Example 2.2. Associated meta knowledge**

\[
\text{G3} \{ \text{G1 mk:source <www.rpi.edu/report.doc>} . \\
\text{G1 mk:timestamp "5/5/2007"} \}
\]

\[
\text{G4} \{ \text{G2 mk:source <www.cs.umd.edu/survey.pdf>} . \\
\text{G2 mk:timestamp "6/6/2001"} \}
\]

In our scenario, the sample user aims to explore the knowledge and meta knowledge using the RDF query language SPARQL. We assume that he aims to find experts in the domain of Semantic Web and their affiliations. The corresponding SPARQL query is shown in example 2.3. In addition, the user wants to exploit meta knowledge from example 2.2 for obtaining results with best certainty and for analyzing contradictory answers (e.g. different affiliations for the same person “James Hendler” in example 2.1).

**Example 2.3. Extracted Knowledge and SPARQL query**

\[
\text{CONSTRUCT \{?x worksAt ?z\}} \\
\text{FROM NAMED G1} \\
\text{FROM NAMED G2} \\
\text{WHERE \{ GRAPH ?g \{?x affiliatedWith ?z . \\n?x researchTopic SemanticWeb}\} }\]

3. DESIGN CHOICES
This section summarizes and shortly motivates the design choices for our meta knowledge framework.

**Reification.** Establishing relationships between knowledge and meta knowledge requires appropriate reification mechanisms for supporting statements about statements. Our general objective is to execute queries on original data (i.e. without meta knowledge) directly, without complex transformations. For compliance with existing applications that access the repository in a common way (e.g. using SPARQL queries), we do not modify existing user data. This requirement does not allow us to use mechanisms like RDF reification, which decompose existing triples and fully change the representational model. In our framework described in section 4, we adopt the notion of Named RDF Graphs for meta knowledge representation [2, 5].

**Storage mechanisms.** Following the overall philosophy of RDF, we do not separate meta knowledge from “normal” user knowledge in the repository. Following this paradigm, a user or developer has unlimited access to all contents of the triple store and can manipulate meta knowledge directly. In other words, the user can directly access meta knowledge (e.g. using suitable SPARQL queries). Beyond explicitly designed queries for meta knowledge access, in Section 5 we describe the extension of SPARQL that allows us to access meta knowledge about the result set automatically without user intervention.

**Dimensions of Meta Knowledge.** An important point for the application design is the definition of relevant meta knowledge properties and their suitable interpretation for arbitrary complex query patterns. In general, these properties are application dependent and must be carefully chosen by the system administrator. In our scenario (sections 2 and 6) we discuss common and widely used properties, such as timestamp, source, and (un)certainty, and show ways of defining and utilizing them in our framework.

**Syntax extensions.** Seamlessly integrated access to meta knowledge requires corresponding extensions of existing querying mechanisms. These can be realized at different levels, for instance at the level of query languages (e.g. SPARQL) or at the level of application-specific interfaces (e.g. Sesame API). In Section 5 we describe our SPARQL extension for constructing query results with associated meta knowledge. It is system-independent and not related to some particular implementation of the RDF repository. Furthermore, it fully supports the existing SPARQL syntax and semantics. Compliance with existing established standards makes the integration with existing applications and interfaces substantially easier.

4. SYNTAX AND SEMANTICS FOR RDF WITH META KNOWLEDGE
In the course of representing and reasoning with meta knowledge we embed a language with meta knowledge reasoning, i.e. RDF*, in a language without such specific facilities, i.e. in RDF. This embedding implies that we may consider an RDF snippet in its literal sense and we may possibly interpret it as making a meta knowledge statement. Embedding meta knowledge in RDF is not the most expressive means to deal with all needs of meta knowledge processing, but it retains upward compatibility with existing usage...
of the language and corresponding tools and methods, which is a major concern for Semantic Web approaches.

Though we denote meta knowledge in RDF, we must distinguish the notation of RDF with only implicit notation of meta knowledge, but no semantic consequences specifically due to this meta knowledge, from a formally extended model of RDF with explicit notation of meta knowledge. The following definition of RDF$^+$ helps us to draw this line very clearly and concisely. The abstract syntax for this extended language, RDF$^+$, is given in Section 4.1 and its semantics in Section 4.2. Eventually in this section, we show how to embed RDF$^+$ in RDF with named graphs.

### 4.1 An Abstract Syntax for RDF$^+$

The abstract syntax of RDF$^+$ is based on the same building blocks as RDF:

- $U$ are Uniform Resource Identifiers (URIs).
- $L$ are all RDF literals.
- $G \subseteq U$ is the set of graph names.
- $P \subseteq U$ is the set of properties.

In addition, we must be able to refer to statements directly without use of reification. For this purpose, we introduce statement identifiers:

- $\Theta$ is a set of statement identifiers, which is disjoint from $U$ and $L$.

Now, we may define RDF$^+$ literal statements that are placed in named graphs and have, in addition to RDF, a globally unique statement identity.

**Definition 4.1.** (RDF$^+$ Literal Statements). The set of all RDF$^+$ literal statements, $\Xi$, is defined as quintuples by:

$$\Xi := \{ (g, s, p, o, \theta) \mid g \in G, s \in U, p \in P, o \in U \cup L, \theta \in \Theta \}.$$

Thereby, $\theta$ and $(g, s, p, o)$ are keys such that there exists a bijection $f_1$ with $f_1(g, s, p, o) = \theta$. Moreover, we define the overloaded function $f_3$ to return the complete quintuple given either $\theta$ or $(g, s, p, o)$, i.e. $f_3(\theta) := (g, s, p, o, \theta) := f_3(g, s, p, o)$, when $f_1(g, s, p, o) = \theta$.

The reader may note that we assume that $f_1$ is fixed and given before any statement is defined. Furthermore, this definition of literal statements and the rest of this paper abstracts from RDF blank nodes in order to keep the formalization more concise. However, we do not see any conceptual problem in extending our treatments to blank notes, too.

The two statements of Graph G1 of Example 2.1 may now be represented in RDF$^+$ in the following way.

---

**Example 4.1.**

$\Xi \supseteq K \supseteq \{ (G1, JamesHendler, researchTopic, SemanticWeb, \theta_1), (G1, JamesHendler, affiliatedWith, RensselaerPI, \theta_2) \}$

Thereby, the exact form of statement identifiers in $\Theta$ is up to the implementation, as they are only used for internal processing.

Having represented the literal interpretation of RDF statements in RDF$^+$, we may now address the representation of selected RDF statements as RDF$^+$ meta knowledge. This is done using a structure of RDF$^+$ meta knowledge statements, $\mathfrak{M}$, that is separate from the set of RDF$^+$ literal statements:

**Definition 4.2.** (RDF$^+$ Meta Knowledge Statements). Let $\Pi \subseteq P$ be the set of meta knowledge properties. Let $\Omega_{\pi}$, with $\pi \in \Pi$, be sets providing possible value ranges for the meta knowledge properties $\pi \in \Pi$.

Then, the set of all RDF$^+$ meta knowledge statements, $\mathfrak{M}$, is defined by:

$$\mathfrak{M} := \{ (\theta, \pi, \omega) \mid \theta \in \Theta, \pi \in \Pi, \omega \in \Omega_{\pi}\}.$$

The following example partially demonstrates the target representation of the first two meta knowledge statements of graph G3 from Example 2.2.

**Example 4.2.**

$$\mathfrak{M} \supseteq M \supseteq \{ (\theta_1, \text{mk:source}, \{<www.rpi.edu/report.doc>\}), (\theta_1, \text{mk:certainty}, 0.9) \}$$

Together we may now define a RDF$^+$ theory.

**Definition 4.3.** (RDF$^+$ Theory). A RDF$^+$ theory of literal statements and associated meta knowledge statements is a pair $(K, M)$ referring to a set of literal statements $K \subseteq \Xi$ and a set of meta knowledge statements $M \subseteq \mathfrak{M}$.

A (partial) example for such a theory is given by the pair $(K, M)$ with definitions for $K$ and $M$ as given in examples 4.1 and 4.2, respectively.

### 4.2 A Semantics for RDF$^+$

We now have an abstract syntax for representing RDF triples like JamesHendler researchTopic SemanticWeb as part of G1 and meta knowledge statements like the source of the statement that James Hendler’s research topic is Semantic Web is found in the document <www.rpil.edu/report.doc>. However, such an abstract syntax may remain remarkably ambiguous if it cannot be linked to a formal semantics. Assume two meta knowledge statements:

$$(\theta_1, \text{mk:source}, \{<www.rpil.edu/draftReport.doc>\})$$

and

$$(\theta_1, \text{mk:source}, \{<www.rpil.edu/finalReport.doc>\})$$

for the same literal statement identified by $\theta_1$, the question may arise whether this means a disjunction, i.e. one of the two documents has provided the fact, or a conjunction, i.e. both documents...
have provided the fact, or a collective reading, i.e. the two documents together gave rise to the fact, or whether this situation constitutes invalid meta knowledge.

In order to prevent such ambiguities we introduce a generic semantic framework for meta knowledge in RDF\(^1\). However, the framework must also be able to reproduce the literal interpretations found in RDF. For the latter purpose, we first define a ‘standard’ model for a RDF\(^1\) theory.

**Definition 4.4 (Standard Interpretation and Model).** A standard interpretation \(I_s : \mathcal{S} \rightarrow \{\top, \bot\}\) for a structure \((K, M)\) assigns truth values to all statements\(^2\) in \(K\).

A standard interpretation is a standard model if and only if it makes all statements in \(K\) become true.

For instance, any standard model \(I_s\) for \((K, M)\) in example 4.1 would include \((G1, JamesHendler, researchTopic, SemanticWeb, \theta_1)\) in its set of literal statements evaluating to \(\top\).

In order to address the level of meta knowledge we foresee an additional model layer that provides a different interpretation to each meta knowledge property.

**Definition 4.5 (Π-Interpretation and Model).** A Π-interpretation \(I_{\pi} : \mathcal{S} \rightarrow \Omega_\pi\) for a property \(\pi \in \Pi\) is a partial function mapping statements into the allowed value range of \(\pi\).

A Π-interpretation \(I_{\pi}\) is a Π-model for \((K, M)\) if and only if for all meta knowledge statements \((\theta, \pi, \omega) \in M\) where \(f_1(\theta) = (g, s, p, o)\) the value of the interpretation coincides with \(\omega\), i.e. \(I_{\pi}((g, s, p, o, \theta)) = \omega\).

As an example, consider the certainty interpretation \(I_{\text{certainty}}\) of the literal statement \((G1, JamesHendler, researchTopic, SemanticWeb, \theta_1)\) from Examples 4.1 and 4.2. A model \(I\) would map this literal statement using \(I_{\text{certainty}}\) onto \(0.9\).

The literal and the meta knowledge interpretations may now be combined to define what an overall, unambiguous model is:

**Definition 4.6 (Meta Knowledge Interpretation and Model).** A meta knowledge interpretation \(I\) is a set including a standard interpretation \(I_s\) and the Π-interpretations \(I_{\pi}\) for all meta knowledge properties \(\pi \in \Pi\).

A meta knowledge interpretation \(I\) is a model for a theory \((K, M)\) if and only if all its interpretations \(I \in \mathcal{S}\) are a standard model or Π-models for \((K, M)\).

### 4.3 Mapping between RDF and RDF\(^1\)

The mapping between RDF and RDF\(^1\) needs to be defined in two directions. First, one must be able to map from RDF as given in the examples from Section 2 to RDF\(^1\). Second, one must be able to map from RDF\(^1\) to RDF. Because RDF\(^1\) is more fine-grained than RDF the first direction will be easy. For the second a compromise on the granularity of the representation has to be made.

\(^1\)Note that because \(f_1\) is fixed there are no two tuples \(\{(g, s, p, o, \theta_1), (g, s, p, o, \theta_2)\}\), where \(\theta_1 \neq \theta_2\). This implies that the standard interpretation is independent of the identifiers \(\theta_1, \theta_2\).

\(^2\)The examples of Section 2 reify groups of statements, i.e. the ones found in \(G1\) and \(G2\), in order to associate meta knowledge, such as given in \(G3\) and \(G4\). In order to allow for an interpretation of the meta knowledge as defined in the preceding section, we map RDF into RDF\(^1\). For all RDF statements, including statements in graphs \(G1\) and \(G2\) of Example 2.1, the mapping performed is close to an identity mapping. One only needs to add statement identifiers. The result for \(G1\) in RDF\(^1\) is:

**Example 4.3.**

\[
K \supseteq \{ (G1, JamesHendler, researchTopic, SemanticWeb, \theta_1), (G1, JamesHendler, affiliatedWith, RensselaerPI, \theta_2) \},
\]

with \(\theta_1 := f_1(G1, JamesHendler, researchTopic, SemanticWeb)\) and \(\theta_2 := f_1(G1, JamesHendler, affiliatedWith, RensselaerPI)\).

The same mapping – close to the identity mapping – is performed for meta knowledge statements like statements of graph \(G3\), resulting in their representation as literal statements:

**Example 4.4.**

\[
K \supseteq \{ (G3, G1, mk:source, <www.rpi.edu/report.doc>, \theta_1), (G3, G1, mk:certainty, “0.9”, \theta_1), \ldots \}
\]

Note that this step is necessary in order to achieve upward and limited downward compatibility between RDF\(^1\) and RDF.

The interpretation of statements, like the ones found in \(G3\), also require an interpretation as meta knowledge. This is achieved by mapping RDF statements with designated properties from \(\Pi\) like \(mk:source\) and \(mk:certainty\) to the additional meta knowledge layer:

**Example 4.5.**

\[
M \supseteq \{ (\theta_1, mk:certainty, “0.9”), (\theta_1, mk:source, <www.rpi.edu/report.doc>), \ldots \}
\]

The mapping of predicates of these meta knowledge statements from RDF to RDF\(^1\) is obvious, they are mapped to itself. Objects are mapped to the corresponding elements of the value ranges \(\Omega_\pi\). For the subjects, however, there arise modeling choices. For instance, if \(mk:certainty\) were interpreted using probability theory, one might assign a distributive or a collective reading. In the distributive reading, each fact in \(G1\) receives the probability value of 0.9 and, eventually, the distributive reading will assign a joint probability of close to 0 for a large number of \(n\) stochastically independent facts, i.e. the joint probability 0.9\(^n\). In the collective reading, the collection of facts in \(G1\) as a whole will receive the probability value 0.9. Therefore, the collective reading will assign an individual certainty close to 1 for each individual fact, when the number of facts is high and each fact is independent from the others, i.e. the individual probability would be 0.9. A priori, none of the two (and more) modeling choices is better than the other, but they constitute different modeling targets.
The mapping from RDF to RDF\textsuperscript{+} for the \textit{distributive reading} of a meta property \(\pi\) is easy to achieve.

**Definition 4.7 (Distributive Embedding).**
Given an RDF statement “\(G \{ S P O \}\)” and an RDF meta knowledge statement “\(H \{ G \pi \omega \}\)” a distributive embedding of RDF\textsuperscript{+} in RDF adds the meta knowledge statement
\[
((\theta, \pi, \omega) \mid \theta = f_1(G, s, p, o) \land \langle \theta, \pi, \omega \rangle \in K)
\]
to \(M\).

This means that such a meta knowledge statement is applied individually to all statements in the graph to which it refers in RDF, as indicated in the example above. For certain \(\pi\) there might be several RDF meta knowledge statements \(H \{ G \pi \omega \}\) which attach different values \(\omega\) to a graph \(G\) via a single meta knowledge property \(\pi\). In that case a set-valued range \(\Omega\) has to be used in order to be consistent with Definition 4.5.

4.3.2 From RDF\textsuperscript{+} to RDF

The serialization of RDF\textsuperscript{+} data in the knowledge base \(K\) is straightforward. Each quintuple \((g, s, p, o, \theta)\) is realized as a corresponding triple in a named graph and the tuple identifier \(\theta\) is discarded.

**Example 4.6.**

\[
(G_5, \text{JamesHendler}, \text{researchTopic}, \text{SemanticWeb}, \theta_1) \quad \text{is mapped to} \quad G_5 \{ \text{JamesHendler researchTopic SemanticWeb} \}
\]

For meta knowledge statements the situation is more challenging, because literal statements with different statement identifiers may belong to only one named graph. Their corresponding meta knowledge statements may differ, but the realization of the meta knowledge statements in RDF does not allow for retaining these fine-grained distinctions – unless one chooses to change the modeling approach drastically, e.g. by assigning each literal statement to a named graph of its own, which seems undesirable (cf. discussion in Section 3).

We have preferred to pursue a more conventional modeling strategy for RDF with named graphs. Therefore, we weaken the association between meta knowledge statements and their corresponding literal statements when mapping to RDF. I.e. we group sets of meta knowledge property values into one complex value.

**Definition 4.8 (Generating Grouped meta knowledge).**

Given an RDF\textsuperscript{+} theory \((K, M)\), RDF meta knowledge is generated by grouping RDF\textsuperscript{+} meta knowledge statements as follows:

Add the triple \((g \pi \omega)\) to the RDF graph \(g' := \text{hashGraph}(g)\) for each
\[
\omega' := \omega_1 \lor \ldots \lor \omega_n,
\]
where \((\theta, \pi, \omega) \in M \land \langle g, s, p, o, \theta \rangle \in K\). Further, hashGraph is a function mapping existing graph names onto graph names suitable for associating meta knowledge and \(\lor\) is an operation defined on \(\Omega\).

If \(\omega'\) is set-valued then a set of triples is added to \(g'\) in order to represent \(\omega'\). The suitability of hashGraph may be application specific. A general strategy may map graph names \(g\) to graph names prefixed by <http://metaknowledge.semanticweb.org> in a deterministic manner. Operations on meta knowledge properties are discussed in section 5.2.

In the following example the grouping of meta knowledge values is illustrated.

**Example 4.7.**

\[
K := \{
(G_5, \text{JamesHendler}, \text{researchTopic}, \text{SemanticWeb}, \theta_1),
(G_6, \text{JamesHendler}, \text{affiliatedWith}, \text{UnivMaryland}, \theta_2) \}
\]

\[
M := \{
\theta_1, \text{mk:source,} \text{http://www.rpi.edu/report.doc}),
\theta_2, \text{mk:source,} \text{http://www.cs.umd.edu/survey.pdf}) \}
\]

is mapped to

\[
G_5 \{ \text{JamesHendler researchTopic SemanticWeb},
\text{JamesHendler affiliatedWith UnivMaryland} \}
G_6 \{ \text{mk:source http://www.rpi.edu/report.doc},
\text{http://www.cs.umd.edu/survey.pdf} \}
\]

In Example 4.7, the resulting grouped value is the set consisting of the two documents <report.doc> and <survey.pdf> which is represented by two triples. For specific meta knowledge properties, an additional function may be necessary to provide a mechanism for representing grouped values in an appropriate RDF data structure.

5. SPARQL FOR RDF AND META KNOWLEDGE

In this section we first introduce a small extension to standard SPARQL syntax [16] and then define how SPARQL can be applied to an RDF\textsuperscript{+} knowledge base. The objective of our considerations is the derivation of meta knowledge about query results.

5.1 SPARQL Syntax Revisited

When using SPARQL to query RDF\textsuperscript{+} we propose only two modifications to obtain meta knowledge. First, we introduce one additional expression “\textit{WITH META MetaList}”. This expression includes the named graphs specified in \textit{MetaList} for treatment as meta knowledge. This statement is optional. When it is present the SPARQL processor may digest the RDF\textsuperscript{+} meta knowledge statements derivable from the RDF named graphs appearing in the \textit{MetaList}. The SPARQL processor will then use this meta knowledge to compute and output all the meta knowledge statements derivable by successful matches of RDF\textsuperscript{+} literal statements with the \textit{WHERE} pattern.

In order to determine which literal statements should be considered we introduce a second modification. We do not process \texttt{FROM} expressions with our meta knowledge framework, but only \texttt{FROM NAMED}. The reason is that \texttt{FROM} \(g\) expressions replicate all RDF triples of \(g\) into the default triple space of the query. Thereby, they remove the links between the RDF statements of \(g\) and possible meta knowledge. Hence, \texttt{FROM} expressions are not relevant for our treatment of meta knowledge, but of course they may still be processed using the standard SPARQL semantics.

Thus, SPARQL queries on RDF\textsuperscript{+} have one of the two following overall forms:
**Definition 5.1** (SPARQL SELECT Query).
The structure of a SPARQL SELECT query has the following form:

\[
\text{SELECT } \text{SelectExpression} \\
\text{(WITH META MetaList)?} \\
\text{(FROM NAMED GraphName)+} \\
\text{WHERE } P
\]

**Definition 5.2** (SPARQL CONSTRUCT Query).
The structure of a SPARQL CONSTRUCT query has the following form:

\[
\text{CONSTRUCT } \text{ConstructExpression} \\
\text{(WITH META MetaList)?} \\
\text{(FROM NAMED GraphName)+} \\
\text{WHERE } P
\]

In these definitions, \( P \) refers to a graph pattern that explains how RDF\(^2 \) literal statements from named graphs specified using FROM NAMED statements are matched. Matches bind variables that are used for providing results according to the SelectExpression or the ConstructExpression.

### 5.2 SPARQL Semantics Revisited

In this subsection we define the semantics of SPARQL queries evaluated on an RDF\(^2 \) theory. For our definitions we use two building blocks: algebraic semantics of SPARQL [12, 14] and the how-provenance calculated via annotated relations (cf. [8]).

The algebraic semantics of SPARQL queries are given based on set-theoretic operations for sets of variable assignments (cf. [12, 14]). Thereby, a variable assignment is a partial function \( \mu : V \rightarrow U \cup L \), where \( V \) is the set of variables given in a SPARQL query. A set of variable assignments can be represented by a relation \( \phi \) over the domain \( (U \cup L)^{\mu} \), where the variables \( V \) are the attributes and assignments are the tuples of this relation. Such a set of assignments may be assigned information about the so-called how-provenance [8], i.e., the assignments may be annotated with formulae describing the individual derivation tree used to assign the variables. The how-provenance annotation may be represented by a function \( \Phi : (U \cup L)^{\mu} \rightarrow \mathfrak{U} \), where \( (U \cup L)^{\mu} \) is the set of all tuples of the length \( |V| \) over the domain \( U \cup L \) and \( \mathfrak{U} \) is the set of formulae annotating variable assignments. The set of formulae \( \mathfrak{U} \) is given by all Boolean formulae constructed over the set of literal statements \( \mathfrak{S} \) and including a bottom element \( \bot \) and a top element \( \top \). The formulae constitute an algebra \( (\mathfrak{U}, \land, \lor, \neg, \top, \bot) \). The special element \( \bot \) is used as annotation of variable assignments which are not in the relation \( \phi \). The special element \( \top \) may be omitted, but it allows for simplification of complex formulae.

Assume the following SPARQL query to be evaluated on the RDF\(^2 \) knowledge base \( K \):

\[
\text{SELECT } \text{SelectExpression} \\
\text{(WITH META MetaList)?} \\
\text{(FROM NAMED GraphName)+} \\
\text{WHERE } P
\]

**Example 5.1.**

\[
\text{SELECT } ?g ?x ?y \\
\text{FROM NAMED G1} \\
\text{FROM NAMED G2} \\
\text{WHERE } \{ \\
\text{GRAPH } ?g \{ ?x \text{ researchTopic } ?y \} \}
\]

**Example 5.2.**

\[
K = \{ \\
(G1, \text{JamesHendler, researchTopic, SemanticWeb, } \theta_1), \\
(G1, \text{JamesHendler, affiliatedWith, RensselaerPI, } \theta_2), \\
(G2, \text{JamesHendler, researchTopic, Robotics, } \theta_3), \\
(G2, \text{JamesHendler, affiliatedWith, UnivMaryland, } \theta_4), \\
(G2, \text{RudiStuder, affiliatedWith, UnivKarlsruhe, } \theta_5) \}
\]

For the query of example 5.1, we may find the following variable assignments using standard SPARQL processing and we may indicate, which atomic formulae, i.e. RDF\(^2 \) quintuples in this simple example, led to these variable assignments. This indication is given by the statement identifiers representing their statements.

**Example 5.3.**

\[
\phi = \{ \\
G1 \text{ JamesHendler SemanticWeb } \theta_1 \\
G2 \text{ JamesHendler Robotics } \theta_3 \\
G2 \text{ RudiStuder SemanticWeb } \theta_5 \\
\}
\]

This simple example of how a set of variable bindings has been produced is generalized to SPARQL queries of arbitrary complexity by a recursive definition of simultaneous query evaluation and computation of the annotations. The first step in evaluating a graph pattern is to find matches for the triple pattern contained in the query. Because the RDF\(^2 \) knowledge base \( K \) consists of quintuples, we need to adapt the SPARQL evaluation procedures. The statement identifiers do not need to be matched, as they depend functionally on graph name, subject, predicate and object. Therefore, we consider matching of quadruple patterns \( (\gamma, \alpha, \beta, \delta) \). As a simplification of our formalization we assume that the keyword GRAPH together with a URI or a graph variable is used in any given SPARQL query. If it is not used, we may expand a given SPARQL query to include it.

**Definition 5.3** (Basic Quadruple Pattern Matching).

Let \( K \) be a knowledge base of RDF\(^2 \) literal statements and \( \mu \) be a variable assignment.

The evaluation of the SPARQL query “GRAPH \( \gamma (\alpha \beta \delta) \)” over \( K \), denoted by \( \langle \text{GRAPH } \gamma (\alpha \beta \delta) \rangle \), is defined by the annotated relation \( \Phi, \text{dom}(\Phi) = \{ \mu \mid \text{dom}(\mu) = \text{vars}(\text{GRAPH } \gamma (\alpha \beta \delta)) \} \):

\[
\Phi(\mu) = \begin{cases} 
\theta & \text{if } \tau(\mu, (\gamma, \alpha, \beta, \delta)) = (g, s, p, o) \land \\
(g, s, p, o, \theta) \in K \land f_1(g, s, p, o) = \theta, \\
\bot & \text{else}
\end{cases}
\]

where \( \text{vars}(P) \) denotes the variables contained in a pattern \( P \) and \( \tau(\mu, (\gamma, \alpha, \beta, \delta)) \) is the quadruple obtained by replacing the variables in \( (\gamma, \alpha, \beta, \delta) \) according to \( \mu \).
An example for this definition is given by evaluating the query from Example 5.1 on the dataset of Example 5.2 delivering the result as indicated in example 5.3.

Basic quadruple pattern matching is not directly applicable, if an expression “GRAPH γ” appears outside a complex triple pattern. In such a case, we first need to distribute the expression “GRAPH γ” appropriately to atomic triple patterns in order to prescribe atomic SPARQL expressions accessible by basic quadruple pattern matching. Because named graphs cannot be nested, this distribution is always possible and unambiguous. In the following we use the function \( \text{quads}(P) \) to denote the query resulting from this transformation. In example 5.4 this transformation is demonstrated on a conjunction of two triple patterns.

**Example 5.4.**

\[
P_1 = \text{GRAPH } ?\text{src} \{  
\begin{align*}  
?x & \text{ researchTopic } ?y, \\
?x & \text{ affiliatedWith } ?z. 
\end{align*} 
\]

\[
\text{quads}(P_1) = \text{GRAPH } ?\text{src} \{  
\begin{align*}  
?x & \text{ researchTopic } ?y, \\
?x & \text{ affiliatedWith } ?z. 
\end{align*} 
\]

Now we define the evaluation of complex graph patterns by operations on sets of variable assignments similar to [12, 14].

**Definition 5.4 (Complex Graph Pattern Matching).** Let \( P_1, P_2 \) be complex graph patterns. The evaluation of graph patterns over \( K \), denoted by \( \llbracket \cdot \rrbracket_K \), is defined recursively:

1. \( \llbracket \text{GRAPH } \gamma (\alpha \beta \delta) \rrbracket_K \) is given by definition 5.3,
2. \( \llbracket \text{GRAPH } P \rrbracket_K = \llbracket \text{quads}(P) \rrbracket_K \),
3. \( \begin{align*}  
\llbracket P_1 \text{ AND } P_2 \rrbracket_K &= \llbracket P_1 \rrbracket_K \cap \llbracket P_2 \rrbracket_K, \\
\llbracket P_1 \text{ OR } P_2 \rrbracket_K &= \llbracket P_1 \rrbracket_K \cup \llbracket P_2 \rrbracket_K, \\
\llbracket P_1 \text{ FILTER } C \rrbracket_K &= \sigma_C(\llbracket P_1 \rrbracket_K). 
\end{align*} \)

The definition uses the operation \( \text{AND} \). In standard SPARQL the operation \( \text{AND} \) is denoted by the absence of an operator. Like [12, 14] we still use the explicit term \( \text{AND} \) in order to facilitate referencing to this operator.

The recursion in the SPARQL query evaluation defined here is indeed identical to [12, 14]. Only the basic pattern matching has been changed slightly. Basic pattern matching now considers quadruples and it annotates variable assignments from basic matches with atomic statements from \( E \) and variable assignments from complex matches with Boolean formulae \( \mathcal{F} \in \mathcal{G} \) over \( S \).

As an example, consider the query from Example 5.5 evaluated on the knowledge base from Example 5.2.

**Example 5.5.**

\[
\begin{align*}  
\text{SELECT } ?\text{hl} & \text{ ?h2 } ?x & ?y \\
\text{FROM NAMED } G1 & \text{ FROM NAMED } G2 \\
\text{WHERE } & \{ \text{GRAPH } ?\text{hl} \{ ?x \text{ affiliatedWith } ?y \} \} \text{ AND} \\
& \{ \text{GRAPH } ?\text{h2} \{ ?x \text{ researchTopic SemanticWeb} \} \} \\
\end{align*} 
\]

Let \( P \) be the graph pattern contained in the \( \text{WHERE} \) clause of the query. Then the evaluation of \( P \) is defined by an algebraic expression:

\[
\llbracket P \rrbracket_K = \llbracket P_1 \text{ AND } P_2 \rrbracket \llbracket ?x = \text{JamesHendler} \rrbracket_K = \sigma_{\text{isJamesHendler}}(\llbracket P_1 \text{ AND } P_2 \rrbracket) = \sigma_{\text{isJamesHendler}}(\llbracket P_1 \rrbracket \llbracket P_2 \rrbracket) = \sigma_{\text{isJamesHendler}}(\Phi_1 \cup \Phi_2)
\]

where \( \Phi_1 \) and \( \Phi_2 \) are relations representing variable assignments and their annotations. In this example and in the preceding definition we have used algebraic operations on sets of annotated bindings. However, we have not yet explained how these operations are used to construct formulas representing the how-provenance. The following definition will specify how complex formulae from \( \mathcal{F} \), which serve as annotations for results of matching complex graph pattern, will be derived.

**Definition 5.5 (Algebra of Annotated Relations).** Let \( \Phi, \Phi_1 \) and \( \Phi_2 \) be sets of annotated variable assignments. We define \( \llbracket \cdot \rrbracket \), \( \cup \) and \( \sigma \) via operations on the annotations of the assignments as following:

- \( \{ \Phi_1 \bowtie \Phi_2 \}(\mu) = \Phi_1(\mu_1) \land \Phi_2(\mu_2) \), where \( \forall x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2) : \mu_1(x) = \mu_2(x) \) and \( \mu = \mu_1 \cup \mu_2 \),
- \( \{ \Phi_1 \cup \Phi_2 \}(\mu) = \Phi_1(\mu) \lor \Phi_2(\mu) \),
- \( \{ \Phi_1 \setminus \Phi_2 \}(\mu) = \Phi_1(\mu) \land \neg \{ \forall x : \Phi_2(\mu)(x) \} \), where \( \forall x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2) : \mu_2(x) = \mu(x) \),
- \( \{ \iota(\Phi) \}(\mu) = \Phi(\mu) \land f_\iota(\mu) \), where \( f_\iota(\mu) \) denotes a function mapping \( \mu \) to either \( \tau \) or \( \bot \) according the condition \( c \),
- \( \{ \Phi_1 \Rightarrow \Phi_2 \}(\mu) = \{ \Phi_1 \bowtie \Phi_2 \}(\mu) \lor \{ \Phi_1 \\setminus \Phi_2 \}(\mu) \).

Let us now continue the evaluation of the query specified in Example 5.5. In order to evaluate the expression \( \sigma_{\text{isJamesHendler}}(\Phi_1 \bowtie \Phi_2) \) we need to determine \( \Phi_1 \) and \( \Phi_2 \) using definition 5.3. The intermediate result is shown in example 5.6. To evaluate the conjunction of two quadruple patterns the operation \( \bowtie \) is applied, the result is shown in example 5.7. The annotation \( \theta_1 \land \theta_2 \) of the first row represents that this assignment has been derived from the conjunction of the two literal statements \( \theta_1 \) and \( \theta_2 \) (see example 5.2). Application of the \( \iota \)-operation to the intermediate results gives the annotated relation shown in example 5.8.
### Example 5.6.

<table>
<thead>
<tr>
<th>Φ1</th>
<th>7h1</th>
<th>7x</th>
<th>7y</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>JamesHendler RensselaerPI</td>
<td>φ2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>JamesHendler UnivMaryland</td>
<td>φ4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>RudiStuder UnivKarlsruhe</td>
<td>φ6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example 5.7.

<table>
<thead>
<tr>
<th>Φ2</th>
<th>7h2</th>
<th>7x</th>
<th>7y</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>JamesHendler</td>
<td>φ1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>RudiStuder</td>
<td>φ3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example 5.8.

\[ \sigma_{\exists x \text{JamesHendler}}(\Phi_1 \land \Phi_2) = \]

<table>
<thead>
<tr>
<th>Φ1</th>
<th>7h1</th>
<th>7x</th>
<th>7y</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>JamesHendler RensselaerPI</td>
<td>(φ1 \land φ2) \land \top</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>JamesHendler UnivMaryland</td>
<td>(φ1 \land φ2) \land \top</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>RudiStuder UnivKarlsruhe</td>
<td>(φ3 \land φ4) \land \top</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The annotations \( \Phi(\mu) \) can now be used to assign truth values for \( \mu \). \( I_\mu \) (see definition 4.4) assigns truth values to all atomic statements \( s_i \in K \subseteq \Sigma \). We extend the interpretation \( I_\mu \) to capture all the Boolean formulae over statements \( \Sigma \).

#### Definition 5.6 (Standard Interpretation of Formulae).

Let \( F, F_1, F_2 \in \Sigma \) be Boolean formulae over \( \Sigma \), let \( F_\mu \in \Sigma \) be an atomic formula. We define the standard interpretation of \( F_\mu \) as follows:

- \( I_\mu(F_\mu) := I_\mu(F_\mu) \)
- \( I_\mu(\neg F) := \bot \) if \( I_\mu(F) = \top \); \( I_\mu(\neg F) := \top \) if \( I_\mu(F) = \bot \)
- \( I_\mu(F_1 \land F_2) = \top \) if \( I_\mu(F_1) = I_\mu(F_2) = \top \), otherwise \( \bot \)
- \( I_\mu(F_1 \lor F_2) = \top \) if \( I_\mu(F_1) = \top \) or \( I_\mu(F_2) = \top \), otherwise \( \bot \)

For instance, \( I_\mu \) returns \( \top \) for the assignment shown in the first row of \( \Phi_1 \land \Phi_2 \) from example 5.7, because the statements \( \theta_1 \land \theta_2 \) are in the knowledge base.

Analogously to \( I_\mu \), we can extend a \( \Pi \)-interpretation \( I_\Pi \) over \( \text{RDF}^+ \) statements to a \( \Pi \)-interpretation \( I_\Pi \) over formulae. Remember that meta knowledge interpretations allow for only one \( \omega \) per \( \theta \in \Theta \) and \( \pi \in \Pi \) (Definition 4.5). In order to make use of the how-provenance represented by the annotations we require that for each meta knowledge property \( \pi \) an algebra \( (\Omega_\pi, \land_\pi, \lor_\pi, \neg_\pi, \tau_\pi, \top_\pi, \bot_\pi) \) with three operations \( \land_\pi, \lor_\pi, \neg_\pi \) and two special elements \( \top_\pi, \bot_\pi \in \Omega_\pi \) is defined. The definition of the algebras can be supplied by a modeler according to the intended semantics of the different meta knowledge properties.

#### Definition 5.7 (II-Interpretation of Formulae).

Let \( F, F_1, F_2 \in \Sigma \) be Boolean formulae over \( \Sigma \), let \( F_\mu \in \Sigma \) be an atomic formula. We define the interpretation \( I_\Pi \) as follows:

- \( I_\Pi(F_\mu) := I_\mu(F_\mu) \)
- \( I_\Pi(\neg F) := \neg I_\Pi(F) \)
- \( I_\Pi(F_1 \land F_2) = I_\Pi(F_1) \land_\Pi I_\Pi(F_2) \)
- \( I_\Pi(F_1 \lor F_2) = I_\Pi(F_1) \lor_\Pi I_\Pi(F_2) \)

For illustration we consider in Example 5.9 the definition of fuzzy logic operations to calculate a possibility measure on variable assignments, operations defined on timestamps which calculate the time of the last modification, and set operations defined for source documents that construct the combined provenance.

#### Example 5.9.

\[
I_{\text{certainty}}(x_1 \land x_2) = \min(I_{\text{certainty}}(x_1), I_{\text{certainty}}(x_2))
\]

\[
I_{\text{certainty}}(x_1 \lor x_2) = \max(I_{\text{certainty}}(x_1), I_{\text{certainty}}(x_2))
\]

\[
I_{\text{certainty}}(\neg x_1) = 1 - I_{\text{certainty}}(x_1)
\]

\[
\Omega_{\text{certainty}} = [0, 1]
\]

\[
I_{\text{time}}(x_1 \land x_2) = \min(I_{\text{time}}(x_1), I_{\text{time}}(x_2))
\]

\[
I_{\text{time}}(x_1 \lor x_2) = \max(I_{\text{time}}(x_1), I_{\text{time}}(x_2))
\]

\[
I_{\text{time}}(\neg x_1) = 0
\]

\[
\Omega_{\text{time}} = [0, \infty]
\]

\[
I_{\text{source}}(x_1 \land x_2) = I_{\text{source}}(x_1) \lor I_{\text{source}}(x_2)
\]

\[
I_{\text{source}}(x_1 \lor x_2) = I_{\text{source}}(x_1) \land I_{\text{source}}(x_2)
\]

\[
I_{\text{source}}(\neg x_1) = \emptyset
\]

\[
\Omega_{\text{source}} = 2^{\Omega_\Pi} D \text{ the set of document URIs}
\]

#### Query forms.

In standard SPARQL query forms, such as SELECT and CONSTRUCT, allow to specify how resulting variable bindings or RDF graphs, respectively, are formed based on the solutions from graph pattern matching [16]. Modifiers, e.g. for projection and ordering, can be applied. The evaluation of SPARQL queries on RDF\(^+\) data differs in that meta knowledge is attached to the results.

The evaluation of SELECT queries on an RDF\(^+\) dataset is based on \( \text{SELECT}(\Phi) \), where \( \Phi \) denotes the set of variables specified in the SelectExpression and \( \text{PROJECT}(\Phi) \) is defined as follows:

#### Definition 5.8 (Projection).

Let \( \Phi \) be a set of annotated variable assignments and \( X \) be a set of variables, then

\[
\text{PROJECT}(\Phi)(\mu) = \left\{ \begin{array}{ll}
\nu & \text{if } \mu \text{ is a partial function defined only on } X, \\
\bot & \text{else}
\end{array} \right.
\]

If \( X \) forms a proper subset of the variables used in the graph pattern then the annotations of all bindings \( \nu \) are aggregated. This aggregation is analog to the generation of grouped meta knowledge described in Definition 4.8. As an example consider the query shown in Example 5.10, which is a slight modification of the query from Example 5.5, applied to the data shown in Example 5.2. For the result see Example 5.11. In contrast to Example 5.7 there is only one row for JamesHendler.
Example 5.10.

\[
\text{SELECT} \ ?x \\
\text{WITH META} \ \text{G1, G4} \\
\text{FROM NAMED} \ \text{G1} \\
\text{FROM NAMED} \ \text{G2} \\
\text{WHERE} \\
\{\text{GRAPH} \ 7h1 \ {?x \ \text{affiliatedWith} \ ?y}\} \ \text{AND} \\
\{\text{GRAPH} \ 7h2 \ {?x \ \text{researchTopic} \ \text{"SemanticWeb"}}\} \\
\}
\]

Example 5.11.

\[
\text{PROJECT}_{G2}(\Phi_1 \ \land \ \Phi_2) = \\
\begin{array}{c|c|c}
\text{JamesHendler} & \text{time} & \text{certainty} \\
\text{RudiStuder} & 8/8/2003 & 0.7 \\
\end{array}
\]

The result of a SELECT query is a set of extended bindings. Such an extended binding contains values for the specified variables and values for each meta knowledge property \(p \in \Pi\) which can be regarded as additional variables. For each binding \(\mu\) these variables \(\pi\) are bound to \(I'_{\text{EXECUTE}}([P]_k(\mu))\), see Example 5.12. For this reason the meta knowledge from Example 5.13 has been used. For instance \(I'_{\text{execute}}((\theta_1 \ \land \ \theta_2) \ \lor \ (\theta_1 \ \land \ \theta_4)) = 0.9\). If no meta knowledge statement \((\theta, \pi, \omega)\) exists for a particular RDF literal statement \(f_i(\theta)\) and a particular meta knowledge property \(\pi\) then \(\exists_{\pi}\) serves as default value. For the result of a SELECT query all bindings from \(\text{PROJECT}_X([P]_k)\) are extended in this way.

Example 5.12.

\[
\begin{array}{c|c|c}
?x & \text{certainty} & \text{time} \\
\text{JamesHendler} & 0.9 & 5/5/2007 \\
\text{RudiStuder} & 0.7 & 8/8/2003 \\
\end{array}
\]

Example 5.13.

\[
M = \\
\{\theta_1, \text{mk:certainty, 0.9}, \theta_1, \text{mk:time, "5/5/2007"}, \theta_2, \text{mk:certainty, 0.9}, \theta_2, \text{mk:time, "5/5/2007"}, \theta_3, \text{mk:certainty, 0.6}, \theta_3, \text{mk:time, "6/6/2001"}, \theta_4, \text{mk:certainty, 0.6}, \theta_4, \text{mk:time, "6/6/2001"}, \theta_5, \text{mk:certainty, 0.6}, \theta_5, \text{mk:time, "6/6/2001"}, \theta_6, \text{mk:certainty, 0.6}, \theta_6, \text{mk:time, "6/6/2001"}, \}
\]

Analogously to standard evaluation, the evaluation of a CONSTRUCT query on an RDF dataset results in a single RDF graph which is built using the graph template specified in the ConstructExpression (see Definition 5.2). This is in line with the fact that the graph template consists of a conjunction of triple patterns and thus quadruple patterns cannot be stated. Similar to the evaluation of SELECT queries the evaluation of CONSTRUCT queries is based on

\[
\text{PROJECT}_X([P]_k), \quad \text{where } X \text{ denotes the set of variables specified in the ConstructExpression. The RDF graph is constructed as described in the following:}
\]

Let \(t_j\) denote triple pattern \(j\) specified in the ConstructExpression, \(P\) denote the graph pattern specified in the WHERE-clause, \((s_{ij}, p_{ij}, o_{ij})\) denote the triple obtained by replacing the variables \(j\) according to a mapping \(\mu\) and \(\hat{\mu}\) denote a new graph name. Then, for each binding \(\mu \in \text{PROJECT}_X([P]_k)\) and for each \(t_j\) the quintuple \((\hat{\mu}, s_{ij}, p_{ij}, o_{ij}, \theta_{ij})\) is added to \(\Xi\), where \(\theta_{ij}\) is the statement identifier \(f_i(\hat{\mu}, s_{ij}, p_{ij}, o_{ij})\). Further \((\theta_{ij}, \pi, \omega_{ij})\) is added to \(\Omega\), where \(\omega_{ij}\) is determined by applying \(I'_{\text{execute}}\) to the formula which annotates the binding. Note that since \(\text{PROJECT}_X([P]_k)\) and the interpretations \(I'_{\text{execute}}\) are functions and further the graph template in ConstructExpression is a set of triples the meta knowledge properties \((\theta_{ij}, \pi, \omega_{ij})\) are unique for a given \(\theta_{ij}\).

As an example for a CONSTRUCT statement consider Example 5.14. Meta knowledge for some of the RDF statements presented in example 5.2 is specified in example 5.13. For graph pattern \(P\) contained in this query the result of \(\text{PROJECT}_X([P]_k)\) is identical to the annotated relation shown in Example 5.7 except for the first two columns. Based on the single triple pattern \(?x \ \text{worksAt} \ ?y\) contained in the graph template and the two bindings contained in \(\text{PROJECT}_X([P]_k)\) two quintuples are constructed and added to the RDF literal statements \(K_{\text{meta}}\) as shown in Example 5.15. \(M_{\text{res}}\) contains the corresponding meta knowledge statements resulting from \(I'_{\text{execute}}([P]_k(\mu))\).

Example 5.14.

\[
\text{CONSTRUCT} \{?x \ \text{worksAt} \ ?y\} \\
\text{WITH META} \ G3, G4 \\
\text{FROM NAMED} \ G1 \\
\text{FROM NAMED} \ G2 \\
\text{WHERE} \\
\{\text{GRAPH} \ 7h1 \ {?x \ \text{affiliatedWith} \ ?y}\} \ \text{AND} \\
\{\text{GRAPH} \ 7h2 \ {?x \ \text{researchTopic} \ \text{"SemanticWeb"}}\} \\
\}
\]

Example 5.15.

\[
K_{\text{meta}} = \\
\{(\text{Gnew1, JamesHendler, worksAt, RensselaerPI, \theta_{new1}}) \} \\
\{(\text{Gnew2, RudiStuder, worksAt, UnivKarlsruhe, \theta_{new2}}) \}
\]

\[
M_{\text{res}} = \\
\{(\theta_{new1}, \text{mk:certainty, 0.9}), \theta_{new1}, \text{mk:time, "5/5/2007"}, \theta_{new2}, \text{mk:certainty, 0.6}, \theta_{new2}, \text{mk:time, "6/6/2001"}, \theta_{new3}, \text{mk:certainty, 0.6}, \theta_{new3}, \text{mk:time, "6/6/2001"}) \}
\]

6. Tasks and Benefits

This section summarizes the discussed steps of meta knowledge representation and utilization for the sample scenario that was introduced in section 2.
6.1 Tasks for the administrator

In order to represent and utilize meta knowledge, the system administrator has to make some design choices. In particular, the application-specific meta knowledge properties must be defined. In our sample scenario, we consider three meta knowledge properties: source, certainty, and timestamp. In the next step, the administrator defines the intended semantics of these properties in order to facilitate query processing with complex expressions and pattern combinations. Using the notion from Section 5.1, we assume that corresponding definitions for meta knowledge properties are defined according to previously discussed Example 5.9.

Finally, data and available associated meta knowledge are represented in RDF using named graphs [2, 5], and imported into our RDF-based repository.

6.2 Processing performed by the System

We assume that the administrator manages the small sample knowledge base introduced in section 2. The knowledge base is transformed into the RDF quintuples shown in Example 5.2 as discussed in section 4. Associated meta knowledge is transformed into further RDF literal statements and RDF meta knowledge statements. For the properties mk:time and mk:certainty the latter are shown in Example 5.13.

Following our sample scenario, the query from Example 2.3 can be reformulated as the query from Example 5.14 which retrieves names of Semantic Web experts together with their affiliations. Internally, the query processor evaluates this query using graph patterns as discussed in 5.1. If P denotes the graph pattern from this query then all matches for all variables in P are given by |P|.

The resulting set of annotated variable assignments is shown in Example 5.7. It contains possible variable assignments, and the how-provenance (A_i) that explains how these source statements have been used.

By combining this information with definitions for meta knowledge properties and available meta knowledge statements, the query processor constructs the result shown in Example 5.15. This result is then serialized in RDF.

6.3 Benefits for the user/developer

The user or application developer can access the knowledge stored in the RDF-based repository in different ways. On one hand, the repository does not change the existing SPARQL semantics and thus fully supports common SPARQL queries. This is an important advantage for compatibility with existing applications and interfaces. On the other hand, the repository supports the advanced SPARQL syntax with metaknowledge support (section 5.1). Thus, the user obtains additional access to valuable meta knowledge that can be used for relevance ranking, conflict resolution, or other applications in connection with retrieved knowledge.

In our application scenario, the user may realize that the query answer is potentially contradictory (James Hendler is affiliated with Rensselaer PI and University of Maryland). By inspecting the associated meta knowledge, he would realize that the second fact was generated by mistake. In fact, it is based on outdated information (knowledge from the document survey.pdf with timestamp 6/6/2001) that was wrongly combined with knowledge from a more recent source (namely document report.doc with timestamp 5/5/2007). It turns out that the affiliation of James Hendler has actually changed from U Maryland to Rensselaer PI, and the erroneous tuple can be safely excluded from further processing.

7. COMPLEXITY

In this section we analyze how the construction of the annotations influences the complexity of the decision problem related to SPARQL. The decision problem associated with the evaluation of a SPARQL query can be stated as follows [12]: Given an RDF dataset D, a graph pattern P and a mapping µ, determine whether µ is in the result of P applied to D. For this decision problem, which we denote by Eval+, an analysis of the complexity is presented in [12, 13]. In the context of RDF datasets and annotated variable assignments we have a slightly different decision problem: Given an RDF dataset D, an RDF graph pattern P, a variable assignment µ and an annotation α determine whether α is the correct annotation of µ. We denote this problem by Eval+. An annotation is correct if the formula is equivalent (in the logical sense) to the formula obtained by evaluating P as defined in section 5.

With the following two theorems we show that for patterns which do not use the OPTIONAL operator Eval+ has the same complexity as Eval. The RDF counterparts of both theorems have been established by [12, 13]. Like [12, 13] we restrict P* to graph pattern which do not contain blank nodes. In the first theorem we consider graph pattern which use only AND and FILTER operations. Bindings obtained by such pattern are annotated with formulae which do not contain any other operator besides ∧ according to definition 5.4.

Theorem 7.1. Eval+ can be solved in time O(|P*| · |D*|) for graph pattern expressions constructed by using only AND and FILTER operators and for annotation formulas using only the operation ∧.

Proof: The proof consists of two parts: In the first part we construct a correct annotation ě for µ and in the second we check whether ě and ě are equivalent. In the following let p_i denote quadruple pattern i from P and r(µ, p_i) denote the quadruple obtained by replacing the variables in the quadruple pattern p_i according to µ.

In order to construct ě we start by evaluating |p_i| for all quadruple pattern using definition 5.3. In order to achieve this D* needs to be searched for all quadruples r(µ, p_i). This can be performed in O(|P*| · |D*|). Then, we construct the algebraic expression ψ by evaluating |p_i| using definition 5.4. This can be performed by traversing P*.

The correct annotation ě of µ in the result of evaluating P on D* is defined as ψ(µ), see definition 5.5. We can construct ě = ψ(µ) in a depth-first traversal of ψ as following:

Let ψ_i, ψ_j be algebraic expressions part of ψ. For each join operation in ψ the annotation of (ψ_i + ψ_j)(µ) is given by ψ_i(µ) ∧ ψ_j(µ). For each select operation σ_c(ψ)(µ) is given by ψ(µ) if condition c is fulfilled otherwise it is given by ⊥. Since traversing P* and ψ each has time complexity O(|P*|) the evaluation of ě = ψ(µ) remains in O(|P*| · |D*|).

Now we determine whether for a given annotation a holds ě = a. We transform both formulas into a normal form in O(|P*| · log |P*|) using associativity, commutativity and idempotency of ∧. First, we remove all brackets then establish an order among atomic formulas.
(identifiers and $\top$, $\bot$) and finally remove duplicates. For $\alpha$ and $\hat{\alpha}$ normalized this way syntactic equality implies logical equivalence. Assuming $|P'| < |D'|$ the overall complexity remains in $O(|P'| \cdot |D'|)$.

**Theorem 7.2.** Eval is NP-complete for graph pattern expressions constructed by using only AND, FILTER and UNION operators.

**Proof:** The proof consists of two parts: In the first part we show that Eval is contained in NP and in the second we establish NP-hardness of Eval. As above, let $p_i$ denote quadruple pattern $i$ from $P'$ and $(\bar{r}_i, p_i)$ denote the quadruple obtained by replacing the variables in the quadruple pattern $p_i$ according to $\mu$.

The first part consists of two steps as well: first we construct a correct annotation $\hat{\alpha}$ for $\mu$ and then we check whether $\alpha$ and $\hat{\alpha}$ are equivalent. In order to construct $\hat{\alpha}$ we start by evaluating $\llbracket p_i \rrbracket$ for all quadruple pattern using definition 5.3. In order to achieve this $D'$ needs to be searched for all quadruples $(\bar{r}_i, p_i)$. Then, we construct the algebraic expression $\psi$ by evaluating $\llbracket P' \rrbracket$ using definition 5.4. This can be performed by traversing $P'$. The correct annotation $\hat{\alpha}$ of $\mu$ in the result of evaluating $P'$ on $D'$ is defined as $\psi(\mu)$, see definition 5.5. We can construct $\hat{\alpha} = \psi(\mu)$ in a depth-first traversal of $\psi$ as following:

Let $\psi_1, \psi_2$ be algebraic expressions part of $\psi$. For each join operation $(\psi_1 \bowtie \psi_2)(\mu)$ is given by $\psi_1(\mu) \land \psi_2(\mu)$. For each union operation $(\psi_1 \cup \psi_2)(\mu)$ is given by $\psi_1(\mu) \lor \psi_2(\mu)$. For each select operation $\sigma(c)(\psi(\mu))$ is given by $\psi(\mu)$ if condition $c$ is fulfilled otherwise it is given by $\perp$.

Time complexity for evaluating $\psi(\mu)$ is $O(|P'| \cdot |D'|)$. The evaluation of $\alpha \equiv \hat{\alpha}$ is more difficult if UNION operations are contained in $P'$. But it is subsumed by checking equivalence of Boolean formulæ which is a NP-complete problem. Thus, the decision problem Eval is contained in NP.

We can deduce that Eval is an NP-complete problem if Eval, which has been shown to be NP-complete [12, 13], can be reduced to it. Eval can be reduced to Eval if the evaluation defined in section 5 results in an annotation $\alpha \equiv \perp$ exactly for such bindings which are not in the result of standard SPARQL evaluation. For graph pattern which do not contain the OPTIONAL operator this can be shown by induction over the structure of algebraic expressions using definition 5.4 and its standard SPARQL counterpart from [12].

8. IMPLEMENTATION

The framework described in this paper has been implemented and is available as an initial prototype. The prototype is available as an open source implementation at http://isweb.uni-koblenz.de/Research/MetaKnowledge together with example queries using artificial data from the LeHigh benchmark.

9. EXPERIMENTS

In order to evaluate the overhead produced by the evaluation of meta knowledge properties for results of SPARQL queries we carried out two experiments based on the well-known LUBM benchmark [9] and our implementation of the meta knowledge framework. Our main aim is to find out whether the evaluation of SPARQL queries remains feasible if provenance and meta knowledge is provided for query results and thus to support the theoretical results of section 7.

A key question is how to separate the additional effort for the evaluation of provenance and meta knowledge from standard SPARQL processing. Triples describing meta knowledge receive an additional meta knowledge interpretation according to section 4. At the same time they are also treated as ordinary RDF triples (and thus can be queried using standard SPARQL). Thus, if we add meta knowledge to a knowledge base this increases its overall size which also increases the workload for standard query processing. In order to account for this we compare query evaluations performed on the same knowledge base which includes meta knowledge. Our implementation is built on top of Sesame 2.0 (beta 6) using query rewriting. The triple store is used to store both, knowledge and meta knowledge. We expect that a native implementation of the meta knowledge framework can achieve an increased performance.

**Query evaluation.** For the evaluation of SPARQL on RDF we defined the results of SELECT queries to be set-valued, see section 5.2. For standard SPARQL [16], however, a SELECT query returns a solution sequence which may contain duplicate elements. All 14 queries of the LUBM benchmark are SELECT queries. To allow for a consistent comparison of extended evaluation on one side and standard evaluation on the other we added the keyword DISTINCT to the queries for standard evaluation. This tells the query processor to eliminate duplicates. Since the evaluation of individual meta knowledge properties can be arbitrarily complex we compare the following three kinds of query evaluation:

- standard evaluation with additional duplicate elimination (SD), as performed by Sesame,
- evaluation of provenance formulæ for each query result (PF) and
- evaluation of four basic meta knowledge properties (M4), namely agent, confidence, creation time and source, see Example 5.9 (we evaluate agent analogously to source).

Only the last type of query processing actually makes use of the additional triples.

**Data.** We added artificial meta knowledge to the LUBM data. Amount and granularity of the additional meta knowledge are key properties of the resulting dataset. We created two datasets containing a different percentage of meta knowledge triples. In the first dataset (MK29) 10 triples each are assigned to a named graph and artificial values of the four meta knowledge properties are attached to it. As a consequence the dataset contains 29 percent of meta knowledge which might be a reasonable scenario in a real world scenario. In the second dataset (MK400) each single triple is assigned to a different graph and the four meta knowledge properties are associated with it. This way the knowledge base contains four times as many meta knowledge triples as knowledge triples.

3available at http://swat.cse.lehigh.edu/projects/lubm/

4www.openrdf.org
The data set MK29 was created based on LUBM OWL data for 10 Universities. We added the meta knowledge by putting groups of 10 consecutive original triples into one named graph and associating random values of the meta knowledge properties agent, confidence, creation time and source with this graph name, see Example 9.1.

Example 9.1.

```xml
<graph>
  <uri>http://www.x-media-project.org/ontologies/someGraph#0_30</uri>
  <triple>
    <uri>http://www.Department0.University0.edu/FullProfessor1</uri>
    <uri>http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#doctoralDegreeFrom</uri>
    <uri>http://www.University882.edu/uri</uri>
  </triple>
  ...
</graph>
```
Table 2: Processing time of query evaluation with and without provenance and meta knowledge for individual queries and the MK29 dataset. Processing times are average values of 10 runs each.

<table>
<thead>
<tr>
<th>Query</th>
<th>Q1</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q10</th>
<th>Q14</th>
<th>Av.</th>
</tr>
</thead>
<tbody>
<tr>
<td># bindings</td>
<td>5</td>
<td>10</td>
<td>411</td>
<td>24019</td>
<td>3</td>
<td>1874</td>
<td>4</td>
<td>75547</td>
<td></td>
</tr>
<tr>
<td>processing time (ms)</td>
<td>SD</td>
<td>127</td>
<td>163</td>
<td>211</td>
<td>357</td>
<td>135</td>
<td>1619</td>
<td>127</td>
<td>977</td>
</tr>
<tr>
<td></td>
<td>PF</td>
<td>324</td>
<td>347</td>
<td>386</td>
<td>995</td>
<td>337</td>
<td>691</td>
<td>325</td>
<td>2320</td>
</tr>
<tr>
<td></td>
<td>M4</td>
<td>340</td>
<td>375</td>
<td>426</td>
<td>2187</td>
<td>346</td>
<td>878</td>
<td>326</td>
<td>10501</td>
</tr>
</tbody>
</table>

Table 3: Processing time of query evaluation with and without provenance and meta knowledge for individual queries and the MK400 dataset. Processing times are average values of 10 runs each.

<table>
<thead>
<tr>
<th>Query</th>
<th>Q1</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q10</th>
<th>Q14</th>
<th>Av.</th>
</tr>
</thead>
<tbody>
<tr>
<td># bindings</td>
<td>5</td>
<td>10</td>
<td>411</td>
<td>6390</td>
<td>3</td>
<td>1874</td>
<td>4</td>
<td>19868</td>
<td></td>
</tr>
<tr>
<td>processing time (ms)</td>
<td>SD</td>
<td>137</td>
<td>189</td>
<td>204</td>
<td>217</td>
<td>131</td>
<td>1596</td>
<td>124</td>
<td>379</td>
</tr>
<tr>
<td></td>
<td>PF</td>
<td>367</td>
<td>350</td>
<td>367</td>
<td>517</td>
<td>316</td>
<td>676</td>
<td>300</td>
<td>769</td>
</tr>
<tr>
<td></td>
<td>M4</td>
<td>346</td>
<td>359</td>
<td>440</td>
<td>859</td>
<td>321</td>
<td>1083</td>
<td>306</td>
<td>1953</td>
</tr>
</tbody>
</table>

The results of the experiments using a random sequence of queries indicate that similar results also hold for real world scenarios. Here caching can be applied by a query processor. A few times evaluations of the same query even were repeated directly one after another in the sequence. The resulting average processing times are smaller but still comparable to the average values for the evaluations of single queries. A key insight is that the overall processing times remain feasible for evaluations on a dataset of up to 1.8 million triples and results of up to 75,000 bindings.

If we compare the processing times obtained for the two different datasets we might expect that for dataset MK400 processing times increase by a larger factor if meta knowledge is involved. MK400 contains a larger number of meta knowledge triples which need to be processed in order to evaluate meta knowledge—especially compared to the number of knowledge triples. However, at least with our implementation, the dominant characteristic of query evaluation appears to be the size of the result set.

10. RELATED WORK

The importance of better understanding the ways by which the result came about is fundamental to many Semantic Web applications and scenarios. The specification of the Semantic Web proof layer was discussed in [11, 15, 10]. Our approach is focused on a different language model (RDF) and provides fine-grained meta knowledge management for retrieval queries with SPARQL that is not directly comparable with proof traces for OWL reasoning.

In the area of database systems, meta knowledge is often represented using an extension of the relational data model, coined annotated relations. Its purpose is primarily the description of data origins (provenance) and the process by which it arrived as a query answer [6, 3, 4, 7]. Basically, our methodology follows the same idea. However, our approach is specially designed for RDF graph models and not directly comparable to metadata models for relational database systems. The same holds for the query language (SPARQL instead of SQL) and its semantics. An important difference to isolated database solutions is the serialization ability of RDF and thus seamless exchanging and utilization of meta knowledge from our framework across the Semantic Web.

Conclusions. At first we consider the experiments with individual queries. From the estimated standard deviations of the 10 runs for each query, kind of evaluation and dataset we conclude that the measurements are reliable enough to draw two general conclusions: On the one hand we can observe a noticeable increase of processing time using our implementation and on the other hand the amount of this increase can be described by a small linear factor.
11. CONCLUSION AND FUTURE WORK

In this paper, we presented an original, generic, formalized and implemented approach for the management of many dimensions of meta knowledge, like source, authorship, certainty, and others, for RDF repositories. Our method re-uses existing RDF modeling possibilities in order to represent meta knowledge. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request meta knowledge without modifying the query proper. We achieve highly flexible and automatically co-ordinated querying for data and meta knowledge, while completely separating the two areas of concern. Our approach remains compatible to existing standards and query languages and can be easily integrated with existing applications and interfaces.

In the future, we will investigate the meta knowledge support for OWL-based knowledge bases with advanced reasoning capabilities. Due to the substantially higher complexity of inferencing and retrieval algorithms (e.g. reasoning in OWL-DL vs. RDF querying with SPARQL) and the distributed nature of knowledge sources in the Semantic Web, the notion of meta knowledge will require further, non-trivial justification. Another interesting research issue is the support for nested meta knowledge (i.e. construction of meta knowledge for the result with respect to additional information about meta knowledge of its origins).

Our long-term objective is the generic, efficient and effective infrastructure for meta knowledge management as an integral part of the proof layer of the Semantic Web.

12. REFERENCES

Bisher erschienen

Arbeitsberichte aus dem Fachbereich Informatik
(http://www.uni-koblenz.de/fb4/publikationen/arbeitsberichte)

Bernhard Schueler, Sergej Sizov, Steffen Staab, Querying for Meta Knowledge,
Arbeitsberichte aus dem Fachbereich Informatik 8/2008

Stefan Stein, Entwicklung einer Architektur für komplexe kontextbezogene Dienste im
mobilen Umfeld, Arbeitsberichte aus dem Fachbereich Informatik 7/2008

Matthias Bohnen, Lina Brühl, Sebastian Bzdak, RoboCup 2008 Mixed Reality League Team
Description, Arbeitsberichte aus dem Fachbereich Informatik 6/2008

Bernhard Beckert, Reiner Hähnle, Tests and Proofs: Papers Presented at the Second
International Conference, TAP 2008, Prato, Italy, April 2008, Arbeitsberichte aus dem
Fachbereich Informatik 5/2008

Klaas Dollschaft, Steffen Staab, Unterstützung und Dokumentation kollaborativer Entwurfs-
und Entscheidungsprozesse, Arbeitsberichte aus dem Fachbereich Informatik 4/2008

Rüdiger Grimm: IT-Sicherheitsmodelle, Arbeitsberichte aus dem Fachbereich Informatik
3/2008

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für
Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik 2/2008

Markus Maron, Kevin Read, Michael Schulze: CAMPUS NEWS – Artificial Intelligence
Methods Combined for an Intelligent Information Network, Arbeitsberichte aus dem
Fachbereich Informatik 1/2008

Lutz Priese, Frank Schmitt, Patrick Sturm, Haojun Wang: BMBF-Verbundprojekt 3D-RETISEG
Abschlussbericht des Labors Bilderkennen der Universität Koblenz-Landau, Arbeitsberichte
aus dem Fachbereich Informatik 26/2007

Algorithmen und Werkzeuge für Petrinetze, Arbeitsberichte aus dem Fachbereich Informatik
25/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS – an Intelligent Bluetooth-

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS - an Information Network for
Pervasive Universities, Arbeitsberichte aus dem Fachbereich Informatik 23/2007

Lutz Priese: Finite Automata on Unranked and Unordered DAGs Extended Version,
Arbeitsberichte aus dem Fachbereich Informatik 22/2007

Mario Schaarschmidt, Harald F.O. von Kortzfleisch: Modularität als alternative Technologie-
und Innovationsstrategie, Arbeitsberichte aus dem Fachbereich Informatik 21/2007

Kurt Lautenbach, Alexander Pinl: Probability Propagation Nets, Arbeitsberichte aus dem
Fachbereich Informatik 20/2007

Rüdiger Grimm, Farid Mehr, Anastasia Meletiadou, Daniel Pähler, Ilka Uerz: SOA-Security,
Arbeitsberichte aus dem Fachbereich Informatik 19/2007
Christoph Wernhard: Tableaux Between Proving, Projection and Compilation, Arbeitsberichte aus dem Fachbereich Informatik 18/2007

Ulrich Furbach, Claudia Obermaier: Knowledge Compilation for Description Logics, Arbeitsberichte aus dem Fachbereich Informatik 17/2007


Rüdiger Grimm, Anastasia Meletiadou: Rollenbasierte Zugriffskontrolle (RBAC) im Gesundheitswesen, Arbeitsberichte aus dem Fachbereich Informatik 15/2007


Ulrich Furbach, Markus Maron, Kevin Read: Location based Informationssystems, Arbeitsberichte aus dem Fachbereich Informatik, 11/2007


„Gelbe Reihe“
(http://www.uni-koblenz.de/fb4/publikationen/gelbereihe)


Kurt Lautenbach, Stephan Philipp, and Alexander Pipl: Bayesian Networks and Petri Nets, Fachberichte Informatik 2-2006
Rainer Gimnich and Andreas Winter: Workshop Software-Reengineering und Services, Fachberichte Informatik 1-2006


Rainer Gimnich, Uwe Kaiser, and Andreas Winter: 2. Workshop "Reengineering Prozesse" – Software Migration, Fachberichte Informatik 15-2005


Reinhold Letz: FTP 2005 – Fifth International Workshop on First-Order Theorem Proving, Fachberichte Informatik 13-2005

Bernhard Beckert: TABLEAUX 2005 – Position Papers and Tutorial Descriptions, Fachberichte Informatik 12-2005

Dietrich Paulus and Detlev Droege: Mixed-reality as a challenge to image understanding and artificial intelligence, Fachberichte Informatik 11-2005


Pascal Hitzler, Carsten Lutz, and Gerd Stumme: Foundational Aspects of Ontologies, Fachberichte Informatik 9-2005

Joachim Baumeister and Dietmar Seipel: Knowledge Engineering and Software Engineering, Fachberichte Informatik 8-2005

Benno Stein and Sven Meier zu Eißen: Proceedings of the Second International Workshop on Text-Based Information Retrieval, Fachberichte Informatik 7-2005

Andreas Winter and Jürgen Ebert: Metamodel-driven Service Interoperability, Fachberichte Informatik 6-2005

Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaaki Kikuchi, and Minoru Asada: Getting closer: How Simulation and Humanoid League can benefit from each other, Fachberichte Informatik 5-2005

Torsten Gipp and Jürgen Ebert: Web Engineering does profit from a Functional Approach, Fachberichte Informatik 4-2005

Oliver Obst, Anita Maas, and Joschka Boedecker: HTN Planning for Flexible Coordination Of Multiagent Team Behavior, Fachberichte Informatik 3-2005

Andreas von Hessling, Thomas Kleemann, and Alex Sinner: Semantic User Profiles and their Applications in a Mobile Environment, Fachberichte Informatik 2-2005