A Mathematical Approach to Ontology Authoring and Documentation

Christoph Lange and Michael Kohlhase

Computer Science, Jacobs University, Bremen, Germany
{ch.lange,m.kohlhase}@jacobs-university.de

Abstract. The semantic web ontology languages RDFS and OWL are widely used but limited in both their expressivity and their support for modularity and integrated documentation. Expressivity, modularity, and documentation of formal knowledge have always been important issues in the MKM community. Therefore, we try to improve these ontology languages by well-tried MKM techniques.

Concretely, we propose embedding the language concepts into OMDoc to make use of its modularity and documentation infrastructure. We show how OMDoc can be made compatible with semantic web ontology languages, focusing on knowledge representation, modular design, documentation, and metadata. We evaluate our technology by re-implementing the Friend-of-a-friend (FOAF) ontology and applying it in a novel metadata framework for technical documents (including ontologies).

1 Introduction

The concept of an "ontology" as a formalization of a shared conceptualization is at the heart of the semantic web – the web of data and intelligent agents. RDFS (RDF Schema/Vocabulary Description Language [BG04]) and OWL (Web Ontology Language [MvH04]), the major semantic web ontology languages, have a limited expressivity: The common OWL sublanguages OWL-Lite and OWL-DL implement two different description logics – decidable subsets of first-order logic [BCM+07]. This was a deliberate design goal as decidability is a prerequisite for web scalability. A common experience in ontology design is, however, that certain axioms in the domains to be modeled exceed the expressivity of the languages chosen for implementation. Sometimes, dumbing down the model to less expressive special cases[1] is sufficient, whereas in other cases, a prose description of the actual axiom is added to the documentation of the ontology.

An example for the latter can be seen in the Friend-of-a-Friend (FOAF) ontology [BM07] for modeling user profiles and simple social relationships: Usually, a *foaf:Group* has members of type *foaf:Agent*, where an agent can be a group, a person, or an organization. The *foaf:membershipClass* property can be used to be more specific about the type of the members of a group by linking an instance

[1] as has, e. g., been done for the DOLCE ontology, a simplified version of which has been formalized in OWL-DL; cf. `http://www.loa-cnr.it/DOLCE.html`

L. Dixon et al. (Eds.): Calculemus/MKM 2009, LNAI 5625, pp. 389–404, 2009.

of *foaf:Group* to an RDFS or OWL *Class*. We can, e. g., require that all members of the KWARC research group in Bremen be computer scientists. Then, if we state that Michael is a member of KWARC, we would like a reasoner to infer that he is a computer scientist, or, vice versa, to complain, if he is classified as a type of person that is not consistent with being one. This combination of ABox and TBox (instance- and terminology-level) reasoning is not supported by OWL reasoners, though. Therefore, *foaf:membershipClass* is not formally described in the OWL-DL implementation of FOAF, but an informal text in the specification explains how application developers can implement hand-crafted support for the missing inference step[2]. Such informal descriptions are often ambiguous[3] and have to be turned into algorithms manually.

In the MKM domain, tensions between high expressivity desired by authors and decidability or even tractability required for web-scalable automated inference are well-known. Earlier, we have discussed the problem of representing expressive mathematical knowledge, such as the theorem that all differentiable functions are continuous and its proof, which involves higher order logic, in semantic web systems [LK08]. This paper proposes a solution by applying well-tried techniques from mathematical knowledge representation to semantic web ontology engineering. We show how the expressive mathematical markup language OMDoc can be used to express and document semantic web ontologies in a way that complies with existing semantic web tools. We also discuss the particular requirement of extensible metadata vocabularies for ontology documentation, which we address by applying our technologies. We evaluate our approach by applying it to FOAF and conclude with a survey of related work and a summary and outlook. This paper is based on [LK09], which provides additional details.

2 Mathematical Semantic Markup with OMDoc

OMDoc [Koh06] is a three-layered semantic markup language for mathematical knowledge. On every layer, the author is free to choose the degree of formality; anything from informal text to shallow annotations to a full formalization (as needed for symbolic computation or automated deduction) is possible. **Objects** can be complex numbers, derivatives, etc. They are usually composed of *symbols* and represented in content markup, using OpenMath [BCC+04] or MathML [W3Ca]. **Statements** are made about objects and model knowledge about our environment in the respective domain. Statement types include model assumptions, their consequences, hypotheses. They have in common that they state relationships between objects and have to be verified or falsified in theories or experiments. A model is fully determined by its assumptions

[2] Note that a way has been found to replace *foaf:membershipClass* by a semantically equivalent OWL-DL-compatible construct using property restrictions [Alf07]. Nevertheless, we keep this as an example as it is easy to understand, the proposed solution has not yet been officially implemented, and is less intuitive for non-experts.

[3] as can be seen in the mail thread following [Alf07].

(also called *axioms*); all consequences are deductively derived from them (via *theorems* and *proofs*); hence, their experimental falsification uncovers false assumptions of the model. **Theories** put symbols and statements into a context. Even the meaning of a single symbol is determined by its context – e. g., the identifier h can stand for the height of a triangle or Planck's quantum of action, and – depending on the current assumptions – a statement can be true or false. While mathematicians fix and describe the context of a statement, these structures have to be modeled explicitly for computer-supported management. For instance, in logic, a theory is the deductive closure of a set of axioms, that is, the (often infinite) set of logical consequences of the model assumptions. Even though, in principle, this fully explains the phenomenon of context, important aspects like the reuse of theories, knowledge inheritance, and the management of theory changes are disregarded completely. Finally, **documents** consist of narrative and content layers. Content layers contain statements or theories, whereas narrative layers sequentially order snippets from content layers. This facilitates the reuse of content from a shared knowledge base (also called "content commons") in documents that are actually consumed by humans: scientific articles, books, or slide shows.

One can easily identify the following correspondences between the semantic web ontology languages RDFS/OWL and OMDoc: **Classes**, **Properties**, and **Individuals** correspond to objects or symbols. **Axioms** and **Rules** correspond to statements, as they state properties of resources. However, a distinction between proper axioms and facts derived from them is not usually made in ontologies. OMDoc, following the "little theories" approach [FGT92], allows for modeling this distinction and thus reducing theories to their core, while still enabling authors to document selected logical consequences of this core within the same theory. **Ontologies** correspond to theories. Both are often designed modularly and import other ontologies or theories. Both entities of an ontology and symbols of an OMDoc theory are identified by URIs (Uniform Resource Identifiers [BLFM05]) within the namespace of the whole theory/ontology.

We claim that OMDoc particularly performs better in *integrated documentation* and *modularity*. It supports mixing formal, semiformal, and informal knowledge in a literate-programming style, and integrating this into documents that can then be adapted to human audiences (cf. Sect. 3.4). As RDFS/OWL axioms could be *reified*, i. e. treated as resources of their own, by giving them a URI, one could in principle attach documentation to all parts of an ontology. In practice, this is supported less well. RDFa as a way of embedding ontologies into XHTML documents [ABMP08] and certain semantic wikis supporting ontology authoring (e. g. IkeWiki [Sch06]) are notable, but to date not yet completely adequate exceptions (cf. Sect. 6 for a discussion of RDFa for ontology authoring). Modularity in semantic web ontologies is optional at best: In RDFS, entities from external ontologies can be reused without restrictions, just by writing down their URIs. This does not make dependencies explicit at all and can easily lead authors into creating inconsistency. If possible at all, one would have to collect all URI references mentioned in an RDFS ontology and then apply some heuristics

to these URIs in order to get hold of the actual ontologies depended upon. OWL improves on this by allowing explicit imports of ontologies via the *owl:imports* declarative – which only permits literal reuse of imported symbols, though. OMDoc greatly enhances modularity by supporting imports via theory morphisms (symbol or formula mappings) and allows for parametric theories. Even literal imports are not yet widely used in web ontologies, and tools usually do not enforce their usage; improvements are to be expected with a more widespread adoption of OWL 2 [CGHM+08]. OMDoc applications rely on proper imports and can already check their consistency; see [RK08] for details.

3 OMDoc as a Semantic Web Ontology Language

OMDoc is XML-based and thus complies with basic web standards like URIs, and any desired logical foundation can be formalized in OMDoc. We can thus make use of the similarities to semantic web ontology languages pointed out above and use OMDoc for modeling ontologies – provided that we overcome certain obstacles, which are addressed in the following subsections: 1. Since OMDoc is uncommitted to a particular logical foundation, it does not have a native understanding of the RDF[4], RDFS, and OWL(-DL) syntax and semantics. Therefore, these foundations have to be modeled as OMDoc meta-theories first. 2. OMDoc theories can import other theories for a modular design, but they cannot directly reference existing semantic web ontologies in order to enhance them. Therefore, we have to specify an import syntax and semantics. 3. OMDoc itself is not supported by any description logic reasoner[5]. Therefore, we need to provide a way to extract semantic web ontologies from theories.

3.1 Knowledge Representation

As a foundation for expressing semantic web ontologies in OMDoc, we wrote theories for RDF, RDFS, and OWL, which declare as symbols all classes, properties, and individuals of these languages. An ontology is then written as follows: Classes, properties, and individuals are declared as *symbols* with a *type*[6]. The type of an object property is, e. g., *owl#ObjectProperty*, i. e. the symbol *ObjectProperty* from our *owl* theory. Class definitions like "Student = Person $\sqcap \geq$ 1 enrolledIn" ("A student is a person, and is enrolled at least once") are given

[4] RDF (Resource Description Framework [RDF04]) is the foundation of knowledge representation on the semantic web. It represents knowledge as a graph, where nodes are instances of *classes* defined in ontologies, edges are instances of *properties*. An edge is usually read as a "subject predicate object" triple.

[5] There are converters from and to the native languages of several common first-order or higher-order theorem provers, though, which demonstrate OMDoc's utility as a mathematical exchange format.

[6] OMDoc has a foundationally unconstrained infrastructure for type systems: objects can be associated with types that are objects themselves. The particular choice of types is only governed by the available theories. Here we define types as part of the RDF, RDFS, and OWL theories.

as OMDoc *definition*s (cf. Listing 1.1[7]). This is a machine-oriented representation that a user would not usually see, but which would render as three lines in Figure 2 and be edited by a tool like the OMDoc-based semantic wiki SWiM [Lan08b, LGP08] using a dedicated formula editor (cf. Fig. 1).

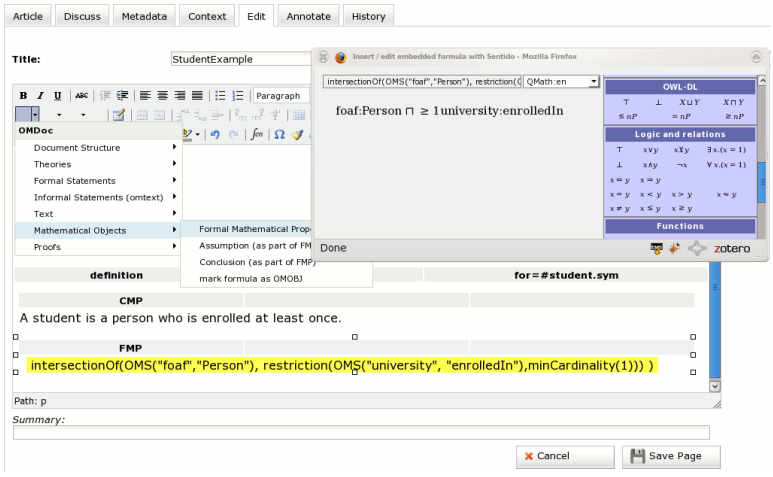

Fig. 1. The definition in SWiM, using the Sentido formula editor. The formula can be edited in OWL abstract syntax [MvH04], or using the tool palette.

Listing 1.1. An OWL ontology in OMDoc: class definition and documentation

```
<theory name="university">
  <imports from="owl.omdoc#owl"/> <!-- The OWL meta-theory -->
  <imports from="foaf.omdoc#foaf"/> <!-- OMDoc wrapper for FOAF -->
  <omtext type="introduction"><CMP>For our "university" ontology, we first import
    FOAF and then introduce the concept of a student. ...</CMP></omtext>
  <symbol name="Student" xml:id="student.sym">
    <metadata>
      <meta property="dc:description">A student</meta>
    </metadata>
    <type system="owl">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
        <OMS cd="owl" name="Class"/></OMOBJ></type>
  </symbol>
  <!-- left out a similar declaration of enrolledIn -->
  <definition for="#student.sym" type="simple">
    <CMP>A student is a person who is enrolled at least once.</CMP>
    <OMOBJ xmlns="http://www.openmath.org/OpenMath">
      <OMA>
        <OMS cd="owl" name="intersectionOf"/>
        <OMS cd="foaf" name="Person"/>
        <OMA>
          <OMS cd="owl" name="Restriction"/>
          <OMS cd="university" name="enrolledIn"/>
          <OMA>
```

[7] `OMS` is the OpenMath syntax for a symbol. `OMA` applies a symbol (usually a function or an operator) to some arguments. `OMI` is an integer.

```
        <OMS cd="owl" name="minCardinality"/>
        <OMI>1</OMI></OMA></OMA></OMA></OMOBJ>
</definition></theory>
```

All other statements can be expressed as OMDoc *axiom*s in such a way that a property is applied to two arguments: a subject and an object. This is the most direct way of representing RDF in OMDoc but does not take advantage of the higher expressivity of OMDoc. However, the author has the possibility to annotate redundant axioms (as introduced in Sect. 2) as *theorem*s instead, which can then be proven on the OMDoc level, using other axioms of the same ontology plus the inference rules of the respective ontology language, as represented in the RDF, RDFS, and OWL theories.

3.2 Connecting OMDoc and Semantic Web URIs

OMDoc and RDF have different ways of giving URIs to symbols. RDF-based ontologies have a namespace URI, which is usually considered to be the URI of the ontology, and all entities within the ontologies have local names. An absolute URI is formed by concatenating the namespace URI and a local name.

OMDoc uses an extended URI-based mechanism for addressing semantic objects. Following the addressing schemes of OpenMath and MathML3, we can address objects by their local name n in their home theory θ, which in turn is referenced by an import path in an OMDoc document identified by a URI g. Thus the URI of a semantic object is of the form $g?\theta?n$; see [RK08] for details. OMDoc allows theory inheritance via renamings – a crucial feature for modularity and ontology interoperability. As a consequence the semantic URIs of OMDoc go beyond traditional URIs and allow to reference objects that are only virtually represented by inheritance.

This difference is largely conventional and does not hinder the integration of OMDoc with RDF-based semantic web ontologies. The only situation where the difference needs to be overcome is where an existing semantic web ontology is rewritten in OMDoc, e. g. for the purpose of documenting it or making its modular structure more explicit, and whenever an OMDoc ontology imports a semantic web ontology. In order to have OMDoc ontologies generate RDF-style URIs, we allow for attaching the namespace URI of the original ontology to a theory via the special metadata field `odo:semWebBase`, which is recognized by our OMDoc→OWL translation presented in the following section. Here is how this would be done for FOAF:

```
<theory name="foaf">
  <metadata>
    <link rel="odo:semWebBase" href="http://xmlns.com/foaf/0.1/"/>
    <meta property="dc:title">Friend of a Friend (FOAF) vocabulary</meta>
  </metadata>
  <!-- imported theories and ontologies left out -->
  <symbol name="Agent"><!-- declaration omitted --></symbol>
  <!-- ... --></theory>
```

This makes sure that the OMDoc→OWL translation gives the *Agent* class its correct URI, i. e. `http://xmlns.com/foaf/0.1/Agent`. We can create an OMDoc theory from a semantic web ontology by simply providing a suitable

`odo:semWebBase` metadata field, only adding symbol declarations, definitions, axioms, etc., later. This is a low-cost way for starting OMDoc-based ontologies which, does not preclude making use of OMDoc's possibilities for documentation and expressive knowledge representation later. Thus we have a suitable migration path from web ontologies to OMDoc.

3.3 Reasoning

Our intention with promoting OMDoc as a more expressive semantic web ontology language is not to replace well-tried technologies for semantic web *reasoning*. While OMDoc does, in principle, allow for alternative approaches to reasoning, being an exchange format for automated theorem provers, this is not the focus of this paper. So in order to allow for writing expressive ontologies in OMDoc while still being able to use optimized reasoners on their tractable/decidable fragments, we defined and implemented a translation from OMDoc to OWL as a module within our Krextor XML→RDF extraction framework [Lan08a]. While the implementation is hard-coded, we aim at giving an exact specification by OMDoc axioms: There is, for example, a set of direct subject–predicate–object axioms (cf. Sect. 3.1) in the OWL theory that state that any application of the *owl#Restriction* symbol to suitable arguments translates to an anonymous RDF resource of type *owl:Restriction* that has certain RDF properties. Extracting RDF triples from OMDoc symbol declarations and axioms is mostly straightforward, but the generation of correct URIs for entities of semantic web ontologies is more involved. We traverse the graph of theory imports and collect the namespace URIs of all theories that carry an `odo:semWebBase` metadatum. Whenever we encounter a reference to a symbol *onto#sym* for an ontology that is implemented as an OMDoc theory *onto*, we generate the semantic web compliant URI as the concatenation of the namespace URI of the theory and the name of the symbol. Here is the RDF generated from the example introduced in Listing 1.1 above[8]:

```
<.../uni.omdoc?university>      rdf:type     owl:Ontology ;
                      owl:imports foaf: .
<.../uni.omdoc?university?Student> rdf:type    owl:Class ;
                          owl:equivalentClass  _:d24e43 .
_:d24e43                  owl:intersectionOf   _:collection-d24e44 .
_:collection-d24e44               rdf:first    foaf:Person ;
                          rdf:rest     _:collection-d24e44-1 .
_:collection-d24e44-1             rdf:first    _:d24e47 ;
                          rdf:rest     rdf:nil .
_:d24e47                  rdf:type     owl:Restriction ;
                      owl:onProperty
                        <.../uni.omdoc?university?enrolledIn> ;
                          owl:minCardinality "1"^^xsd:nonNegativeInteger .
```

The result looks is somewhat illegible (compared e. g. to Fig. 2); in fact there are less technical representations of OWL [HPS08], but in practice it does not

[8] This is Turtle, a text-oriented serialization for the RDF data model. Identifiers prefixed with _ denote anonymous ("blank") nodes that are only accessible within the current RDF graph. The class, which a student is defined to be equivalent to, is represented as a union class of a set of classes, represented as a linked list.

make a difference, as all OWL tools are required to support the RDF representation. Most of the statement- and theory-level structure of OMDoc, such as the distinction between defined and inferred statements and theory morphisms, is lost and uniformly translated to less expressive OWL axioms. Thus, our translation works like a compiler and linker that creates (OWL/RDF) object code from a higher-level OMDoc source code.

3.4 Documentation and Presentation

OMDoc comes with an elaborate, adaptive presentation framework for creating human-readable documents from semantic markup [KMR08]. Mathematical formulae are rendered as Presentation MathML; structures on the statement and theory levels, and complete documents, are rendered as XHTML. For every mathematical symbol, one or more *notation*s can be defined – compare, e. g., our initial OWL example in the German DL notation (Student = Person $\sqcap \geq$ 1 enrolledIn) vs. the Manchester syntax [HPS08]:

```
Class: Student
  EquivalentTo: Person that enrolledIn min 1
```

A default notation is usually provided by the author of a theory, but users can also author their own ones to customize the presentation to their preferences. Initially, the renderer collects all available notation definitions from all imported theories. For every symbol in a content formula as the one in Listing 1.1, the renderer selects from those notation definitions that match the symbol the most appropriate one for the current presentation context, which is made up of, e. g., the language of the enclosing document, the domain of application, or user preferences. The output is parallel markup [W3Ca, section 5.4], which allows for implementing additional services that facilitate browsing and reading – for example linking rendered symbols to the place where they are introduced. A reader who does not know, e. g., the symbol $\sqcap$ in our sample formula, can click on it and thus navigate to the section of the document rendered from the *owl* OMDoc theory that declares (and documents!) the symbol *owl:intersectionOf*. We have implemented this in SWiM using XLinks; the JOBAD active document framework even displays definitions as tooltips without forcing the user to leave the document [GLR09]. Documentation can be given in metadata blocks (cf. Sect. 4), which can be attached to any element on the statement and theory level (cf. Listing 1.1). Textbook or literate-programming style is also possible: A theory can not only contain formal statements but also informal text sections, and *definition*s, *axiom*s, and *theorem*s can have both formal and informal content (*CMP* and *FMP*; cf. Listing 1.1).

4 Scalable Metadata for Technical Specifications

In the previous sections, we have already used metadata for documenting ontologies. Simple metadata vocabularies like Dublin Core (DC [Dub08]) or Creative Commons licensing information (CC [AALY08]) are suitable for retrieval, e. g.

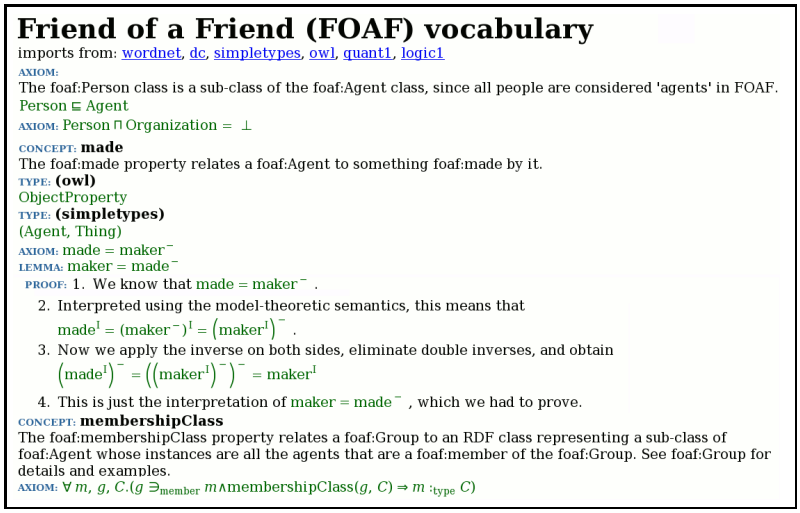

Fig. 2. FOAF in OMDoc, rendered (slightly shortened cf. Sect. 5). We defined some custom notations, e. g. rendering *foaf:member* like set membership, and we combined domain and range of a property into a "relation type".

using a search engine. Specialized document and knowledge management tasks require more complex metadata. In practical application scenarios where OMDoc is used to author formal specifications of safe and secure technical devices, we have particularly experienced a need for documenting the change history of a formal document within that document. Note that a revision log within a document is not intended to replace a versioned repository on the server side – which we also use for OMDoc –, but as an extension for certain use cases. Sometimes, for example, a persistent revision log is required for legal reasons.

4.1 Metadata in OMDoc 1.2

OMDoc allows for attaching a metadata record to any element on the document, theory, and statement level [Koh06, chapter 12]. The current version 1.2 provides XML syntax for all DC and CC properties, plus a few extensions[9], most notably a simple vocabulary for recording revision histories, which has been added to the `dc:date` XML element: The additional `who` attribute refers to the URI of a `dc:creator` or `dc:contributor` element in the same metadata record, and the `action` attribute can have values like "updated", "created", or "imported".

This way of representing metadata has various drawbacks: The vocabulary is hard-coded and not extensible. There is no easy way of adding other vocabularies to OMDoc. Secondly, OMDoc is not aware of the formal semantics of these vocabularies. They have been integrated into the *syntax* of OMDoc, but their

[9] Here, we only give a short summary. Please see the OMDoc 1.2 specification [Koh06] and the extended version of this paper [LK09] for details.

semantics is only available informally as a part of the natural-language specification of OMDoc [Koh06, chap. 12]. More formal semantics for DC and CC would be available as RDFS ontologies, but those have not been incorporated into OMDoc. Even worse, OMDoc's DC extension for revision histories does not have any formal semantics at all. This lack of formal semantics has restricted the attractivity of OMDoc's metadata for application developers. So far, support for them has not been implemented by any OMDoc-aware application, except our own semantic wiki SWiM [Lan08b] and the e-learning environment ActiveMath [GUM+04]. ActiveMath makes use of additional vocabularies for educational metadata, but they are hard-coded into the XML schema in an even less extensible way than in OMDoc 1.2, as they are not distinguished by different namespaces [GUM+04] (ActiveMath's document format forked off OMDoc1.1.).

4.2 The New Metadata Framework

Requirements for a new metadata framework for OMDoc were as follows:

1. Stay backwards-compatible with OMDoc 1.2 concerning expressivity. That is, continue supporting DC and CC, and the custom extensions.
2. Make the formal semantics of vocabularies available to OMDoc applications.
3. Incorporate vocabularies for versioning (for technical documents in particular) and people (for bibliographical data).
4. Don't hard-code a fixed set of vocabularies into the language but stay flexible and extensible for many applications, even future and unknown ones.

Given the fact that many existing metadata vocabularies, including DC and CC, have an RDF semantics, and that with RDFa [ABMP08] a standard for flexibly embedding metadata into X(HT)ML documents had recently stabilized, we chose to incorporate a subset of RDFa into OMDoc, and to look for RDF-compatible metadata vocabularies to satisfy our further requirements. So far, RDFa has only been specified for the "host languages" XHTML and SVG (cf. [W3Cb]), but the specification foresees the integration into other XML-based languages. The new metadata framework introduces the elements `meta` and `link` with the same semantics as their XHTML counterparts as children of any `metadata` block. Resources with document-local identifiers only, i. e. *blank nodes*, can be created using the `resource` element:

Element	Attributes	Children
meta	property, content, datatype	literal text or XML (optional)
link	rel, rev, href	(resource|meta|link)*
resource	about, typeof	(meta|link)*

Due to the inherent flexibility of RDFa, any metadata vocabulary can be used. However, we give particular recommendations for metadata in the above-mentioned domains of special interest. Using DC and CC metadata with the new RDFa syntax for OMDoc is trivial. Our previous DC extensions for revision logs were not immediately RDF-compatible, as they were given as additional

annotations to triples, and no formal semantics was defined for them. Therefore, we replaced them by a completely re-engineered versioning ontology. This ontology reuses the core of the ModelDriven.org versioning ontology [Mod08], with classes *DataAsset* (of which anything on the statement, theory, or document level of OMDoc is a subclass), *Revision*, and *Change*, where an *DataAsset* has *Revision*s, and a *Change* represents a transition from one *Revision* to the following one. As we made *Change* a subclass of the *Event* class from the event ontology [RA07], a change can have a date and an agent. Instead of a generic *Change*, a more specific subclass can be chosen. In future, we plan to introduce specific change types (e. g. for adding a type declaration to a symbol), in a similar way as the OMV Ontology Metadata Vocabulary does for semantic web ontologies [HPHGP07].

Here is a part of the metadata block of a digital library edition of Fermat's last theorem that documents the revision history. The resource has two revisions; for each, the act of creation has an author and a date given as additional metadata:

```
<link rel="rev:created_by_act" href="[_:creation]"/>
<link rel="rev:current_version" href="[_:current]"/>
<link rel="rev:has_version">
  <resource about="[_:v1]" typeof="rev:Revision">
    <link rel="rev:content" href="fermats-last-theorem?rev=1"/>
    <link rel="rev:created_by_act">
      <resource about="[_:creation]" typeof="chg:Creation">
        <link rel="event:agent" href=".../Pierre_de_Fermat"/>
        <meta property="dc:date">1637-06-13T00:00:00</meta>
      </resource></link></resource></link>
<!-- revision 2 (proof by Wiles) omitted to save space -->
<link rel="rev:has_version">
  <resource about="[_:current]" typeof="rev:Revision">
    <link rel="rev:content" href="fermats-last-theorem?rev=3"/>
    <link rel="rev:created_by_act">
      <resource typeof="chg:Import">
        <link rel="event:agent" href="http://.../kohlhase"/>
        <meta property="dc:date">2006-08-28T00:00:00</meta>
        <link rel="rev:prior_version" href="[_:v2]"/>
      </resource></link></resource></link>
```

As we modeled our metadata ontologies in OMDoc, we are now able to extend it by a formal specification of certain rules that had only informally been stated in the OMDoc 1.2 specification: for example, that most DC metadata propagate from document sections down into subsections unless subsections specify different values, or that any *dc:creator* of a subsection of a document becomes a *dc:contributor* to the whole document.

4.3 Extracting Metadata to RDF

Similarly to the extraction of RDF representations of OWL ontologies written in OMDoc (cf. Sect. 3.3), we implemented a Krextor extraction module for RDFa. We then divided the RDFa extraction rules into XHTML-specific ones and into generic ones, the latter of which we combined with support for our OMDoc-specific metadata syntax. The extraction of RDFa from OMDoc is performed both in the extraction of OWL from OMDoc, where it enriches the extracted ontologies with metadata, and in the extraction of RDF outlines from OMDoc in terms of the OMDoc's own document ontology. The latter is a foundation

for semantic web applications having OMDoc (and not OWL) as their native language, such as the semantic wiki SWiM [Lan08b].

4.4 Annotation

As the listing in Sect. 4.2 shows, the new RDFa-based metadata syntax is much more verbose than the old one of OMDoc 1.2. Therefore, we suggest two ways of facilitating the annotation: For manual authoring, we keep the old, "pragmatic" OMDoc 1.2 syntax and specify a transformation of such annotations to the new, "strict" RDFa syntax – implementable, e. g., in XSLT. Having a rich pragmatic syntax that is convenient to author and a strict syntax that is more suited for automated processing and validation is actually a general strategy that we first introduced in MathML 3 and also employ for other aspects of OMDoc. In certain application settings, we can generate part of the metadata automatically. In the SWiM wiki [Lan08b], for example, the names of the author and the contributors of a document are known from the user profiles of these persons and only inserted into the metadata record of a document when it is exported from the wiki to a file. The same holds for the revision history.

5 Evaluation and Discussion

We evaluated our approach on a reimplementation of FOAF in OMDoc. From studying the OWL implementation and the specification of FOAF, we noticed the following problems, which we were able to solve using OMDoc:

1. FOAF references entities from other ontologies (DC, WordNet, Geo Positioning, etc.), but it does not import them. With OMDoc tools (as described in [RK08]), we can identify imports missing in an OMDoc ontology, and our OMDoc→OWL translation (Sect. 3.3) adds them to the OWL ontology resulting from the translation.
2. The source code contains notes for developers as XML comments. In the OMDoc version of FOAF, we were instead able to create informal text sections for them. Other XML comments divide the ontology into sections like "naming properties". In OMDoc, we were able to model document sections without disrupting the logical structure of the ontology.
3. Some of these comments were attached to individual triples, e. g. *foaf:mbox_sha1sum rdf:type owl:DatatypeProperty*. Thanks to literate programming in OMDoc, we could precisely add them as informal comments (*CMP*s) to the respective OMDoc statements.
4. The following properties are inverses of each other: *foaf:maker* $=$ *foaf:made*$^-$, *foaf:depiction* $=$ *foaf:depiction*$^-$, *foaf:topic* $=$ *foaf:page*$^-$, and *foaf:primaryTopic* $=$ *foaf:isPrimaryTopicOf*$^-$. While for each $p = q^-$, an OWL reasoner can infer $q = p^-$, using its built-in axioms for DL reasoning, FOAF redundantly declares each inverse relationship for both participating properties for the purpose of documentation. OMDoc allows for making the difference explicit: For any of the above p, q property pairs, we picked one p and stated $p = q^-$ as an axiom, but $q = p^-$ as an *assertion* that can (provably) be derived from the axiom and the semantics of *owl:inverseOf*, as shown in Fig. 2. Domain and range of inverse properties can be handled similarly.

5. We were able to express the non-OWL semantics of *foaf:membershipClass* (cf. Sect. 1). We chose the first-order-logic representation shown in Fig. 2.
6. The correspondence of *foaf:maker* to *dc:creator* is only defined in prose. The specification suggests using *foaf:maker* whenever the agent who created something is known by URI, and to use the less semantic *dc:creator*, which neither has range nor domain declared, when the creator is only known by a string. Then, it also informally states a rule that the *foaf:name* or *rdfs:label* of the *foaf:maker* of something is the same as the *dc:creator* of that thing. The rule can be captured by a first-order-logic expression in OMDoc, or alternatively by an OWL 2 property chain inclusion [CGHM$^+$08]. The notion that *foaf:maker* is similar to *dc:creator* but has a stronger semantics can be captured by having the FOAF theory import the DC theory and defining a *view* on DC, namely a morphism that maps *dc:creator* to *foaf:maker*. Views frequently occur in mathematics. We can, for example, model the theory of integers by a view $\{\circ \mapsto +, e \mapsto 0\}$ on the theory of monoids, where $\circ$ is the binary operation and e the unit element of the monoid.
7. Finally, we were able to include the informal sections and descriptions of the FOAF specification [BM07] right into the ontology document. This allows for a unified management of the formal specification and its informal explanation, including the introductory chapters and the change log, in a single, coherent document, of which both OWL and XHTML can be generated. The original FOAF specification is generated from the OWL ontology and a set of HTML snippet files with detailed informal descriptions as input using a script, a FOAF-independent version of which is also available [Boj].

This enhanced expressivity of the OMDoc implementation comes at the expense of much more verbosity. While in RDF one can easily attach another axiom to a class (stating, e. g., a subclass relationship or disjointness), most of these triples have to be represented as a individual axiom in OMDoc, unless there is an intuitive way of capturing their semantics as types. While better annotation tools could help (cf. Sect. 4.4), there is also a mathematical approach to improving this: One could add additional axioms to the OMDoc theory for OWL, which introduce operators for shorthand notations (such as pairwise disjointness of a whole set of classes) that imply multiple atomic statements– but then all these axioms would have to be *applied* before generating OWL from OMDoc. This can be done by supporting λ-calculus at the meta level and β-reducing all OMDoc axioms before generating OWL.

As the new metadata framework has not yet been deployed to the OMDoc users, our evaluation focused on the coverage of the RDFa extraction. We first implemented XHTML+RDFa support and then generalized that, so we could evaluate our implementation against the W3C RDFa test suite [HY07], of which it currently passes 90 out of 100 test cases.

6 Related Work

Concerning **expressive ontologies**, the Common Algebraic Specification Language (CASL) and its extensions for various logics are related to OMDoc and its module system. For the CASL-based Heterogeneous Tool Set (Hets), it has been investigated how to integrate OWL-DL and more expressive logics within

a logical framework [KLMN08]. However, CASL is a purely formal language and does not allow for integrating documentation. Concerning **integrated ontology documentation**, RDFa in combination with RDF-based ontologies is similar to our approach. RDFa has mainly been used for ABox knowledge so far; we are only aware of one application of RDFa for TBox knowledge: Ontology Online is a web site for browsing and querying OWL and RDFS ontologies. Every page visualizes one entity of an ontology – as XHTML with the original OWL or RDFS embedded as RDFa annotations [Dec07]. **Metadata for technical specifications** are supported by DocBook [Wal08], a semantic markup language that had originally been conceived for software documentation. DocBook has hard-coded, non-extensible markup for metadata, covering general DC-like metadata, revision histories, and more. None of this has an RDF semantics. There is, however, a workaround for adding RDF-compatible annotations to DocBook: Any DocBook element can carry XLink attributes, from which RDF can be harvested [Dan00].

7 Conclusion and Further Work

By connecting the semantic markup language OMDoc and techniques from MKM to the semantic web standards OWL and RDFa, we contributed a language for ontologies and technical specifications that supports different levels of expressivity and formality but still remains compatible with the existing semantic web infrastructure. As an anonymous reviewer pointed out, we use (MK)M – i. e. techniques from the management of mathematical knowledge – for M(KM) – i. e. for managing knowledge in a mathematical way. We consider our scalable metadata framework applicable to other MKM languages, such as OpenMath CDs, as well. For example, it has been proposed to add a metadata field for the author to OpenMath 3 content dictionaries[10]. Simply employing RDFa with an appropriate ontology would facilitate such decisions.

Our next planned step is identifying further possibilities to modularize the ontologies that we have implemented in OMDoc so far – including the OMDoc formalizations of RDF, RDFS, and OWL –, or making existing modularity more explicit, and then integrating our OMDoc→OWL translation with Hets to enable heterogeneous reasoning [KLMN08][11]. Finally, we want to apply existing OMDoc applications to ontologies written in OMDoc, enhance the SWiM wiki by user interface elements for more conveniently editing and browsing such ontologies, and evaluate its usability in a case study involving ontology engineers. Our group is working on a distributed database for mathematical documents (TNTBase [Zho09]). This database will also employ our document renderer and then follow the practice of content negotiation that is well established on the semantic web [SC08]: OMDoc-aware clients will get OMDoc, semantic web clients will get extracted RDF, and web browsers will get XHTML+MathML – a foundation for a *mathematical semantic web*.

[10] See `http://trac.mathweb.org/OM3/ticket/12`

[11] See `http://trac.kwarc.info/krextor/roadmap` for work in progress.

Acknowledgments. We would like to thank Florian Rabe for help with modeling RDF, RDFS, and OWL as OMDoc theories, Siarhei Kuryla for his contributions to the implementation, and Richard Cyganiak for sharing insights about RDFa. This work was supported by JEM-Thematic-Network ECP-038208.

References

[AALY08] Abelson, H., Adida, B., Linksvayer, M., Yergler, N.: ccREL: The Creative Commons Rights Expression Language. Technical report, Creative Commons (2008), http://wiki.creativecommons.org/Image:Ccrel-1.0.pdf

[ABMP08] Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and processing. Recommendation, W3C (2008)

[Alf07] Alford, R.: Proposal: Deprecate membershipClass, add memberOf. E-mail (2007), http://lists.foaf-project.org/pipermail/foaf-dev/2007-May/008551.html

[BCC+04] Buswell, St., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaetano, M., Kohlhase, M.: The Open Math standard, version 2.0. Technical report, Open Math Society (2004)

[BCM+07] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press, Cambridge (2007)

[BG04] Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema. Recommendation, W3C (2004)

[BLFM05] Berners-Lee, T., Fielding, R., Masinter, L.: Uniform resource identifier (URI): Generic syntax. RFC 3986, IETF (2005)

[BM07] Brickley, D., Miller, L.: FOAF vocabulary specification 0.91. Technical report, ILRT (2007)

[Boj] Bojārs, U.: SpecGen 4 – ontology specification generator for RDFS and OWL, http://code.google.com/p/specgen

[CGHM+08] Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web 6(4) (2008)

[Dan00] Daniel Jr., R.: Harvesting RDF statements from XLinks. Note, W3C (2000)

[Dec07] Decraene, D.: Online ontology visualisation: Embedding OWL-RDFS syntax in XHTML with RDFa (2007), http://ontologyonline.blogspot.com/2007/11/embedding-owl-rdfs-syntax-in-xhtml-with.html

[Dub08] Dublin Core metadata element set, version 1.1. DCMI (2008)

[FGT92] Farmer, W., Guttman, J., Thayer, X.: Little theories. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607. Springer, Heidelberg (1992)

[GLR09] Giceva, J., Lange, C., Rabe, F.: Integrating web services into active mathematical documents. In: MKM/Calculemus 2009 Proceedings. LNCS (LNAI). Springer, Heidelberg (in press, 2009)

[GUM+04] Goguadze, G., Ullrich, C., Melis, E., Siekmann, J., Gross, C., Morales, R.: LeActiveMath Structure and Metadata Model. Deliverable D6 (2004)

[HPHGP07] Hartmann, J., Palma, R., Haase, P., Gómez-Pérez, A.: Ontology Metadata Vocabulary – OMV (2007), http://omv.ontoware.org

[HPS08] Horridge, M., Patel-Schneider, P.F.: OWL 2 web ontology language: Manchester syntax. Working draft, W3C (2008)

[HY07] Hausenblas, M., Yung, W.C.: RDFa test suite. Editor's Draft, W3C (2007), http://www.w3.org/2006/07/SWD/RDFa/testsuite/

[KLMN08] Kutz, O., Lücke, D., Mossakowski, T., Normann, I.: The OWL in the CASL – designing ontologies across logics. In: Sattler, U., Dolbear, C., Ruttenberg, A. (eds.) OWL: Experiences and Directions (2008)

[KMR08] Kohlhase, M., Müller, C., Rabe, F.: Notations for living mathematical documents. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) AISC 2008, Calculemus 2008, and MKM 2008. LNCS (LNAI), vol. 5144, pp. 504–519. Springer, Heidelberg (2008)

[Koh06] Kohlhase, M.: OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]. LNCS (LNAI), vol. 4180. Springer, Heidelberg (2006)

[Lan08a] Lange, C.: Krextor – the KWARC RDF extractor (2008), http://kwarc.info/projects/krextor/

[Lan08b] Lange, C.: SWiM – a semantic wiki for mathematical knowledge management. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 832–837. Springer, Heidelberg (2008)

[LGP08] Lange, C., González Palomo, A.: Easily editing and browsing complex OpenMath markup with SWiM. In: Libbrecht, P. (ed.) Mathematical User Interfaces Workshop (2008)

[LK08] Lange, C., Kohlhase, M.: A Semantic Wiki for Mathematical Knowledge Management. In: Rech, J., Decker, B., Ras, E. (eds.) Emerging Technologies for Semantic Work Environments. IGI Global (2008)

[LK09] Lange, C., Kohlhase, M.: A mathematical approach to ontology authoring and documentation (2009), https://svn.omdoc.org/repos/omdoc/trunk/doc/blue/foaf/note.pdf

[Mod08] ModelDriven.org versioning ontology (2008), http://modeldriven.org/2008/ArchitectureOntology/doc/Versioning.html

[MvH04] McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. Recommendation, W3C (2004)

[RA07] Raimond, Y., Abdallah, S.: The event ontology. Technical report (2007), http://motools.sourceforge.net/event/

[RDF04] Resource description framework (RDF) (2004), http://www.w3.org/RDF/

[RK08] Rabe, F., Kohlhase, M.: An exchange format for modular knowledge. In: Rudnicki, P., Sutcliffe, G. (eds.) Knowledge Exchange: Automated Provers and Proof Assistants (KEAPPA) (2008)

[SC08] Sauermann, L., Cyganiak, R.: Cool URIs for the semantic web. Working Draft, W3C (2008)

[Sch06] Schaffert, S.: IkeWiki: A semantic wiki for collaborative knowledge management. In: 1st Workshop on Semantic Technologies in Collaborative Applications (STICA)(2006)

[W3Ca] W3C. Mathematical Markup Language (MathML) 3.0, 3rd edn.

[W3Cb] W3C. Scalable Vector Graphics (SVG) Tiny 1.2

[Wal08] Walsh, N.: DocBook 5.0: The Definitive Guide. O'Reilly, Sebastopol (2008)

[Zho09] Zholudev, V.: TNTBase (2009), https://trac.mathweb.org/tntbase/