JACOBS
UNIVERSITY

Matthias Bröcheler

A Mathematical Semantic Web

Technical Report No. 15
March 2008

School of Engineering and Science

A Mathematical Semantic Web

Matthias Bröcheler

School of Engineering and Science
Jacobs University Bremen gGmbH
Campus Ring 6
28759 Bremen
Germany

E-Mail: mbroechele@jacobs-alumni.de
http://www.jacobs-university.de/

Summary

The body of mathematical knowledge is rapidly increasing and constantly changing. Zentralblatt MATH[1], an abstracting and reviewing service in the field of mathematics, maintains a database of more than 1.6 million mathematical documents and reports an annual growth by 80,000 articles [zen,]. Similarly, the open internet archive for for electronic preprints of scientific papers, arXiv.org[2], contains close to half a million documents. These figures suggest that neither a mathematician's memory nor the time she devotes to studying new publications can possibly suffice to cover a significant fraction of the accumulated wealth of mathematical knowledge. In addition to the exponential increase in the amount of information, one also observes an increase in the complexity of mathematical content with more interdependencies between different areas within and beyond mathematics.

We propose a Mathematical Semantic Web to support mathematicians in efficiently managing and retrieving mathematical knowledge using computer systems and the internet. As with the World Wide Web, the Mathematical Semantic Web will allow authors of mathematics to publish their documents online which cumulate to a gigantic, decentralized and dynamic mathematical knowledge base. Authors semantically annotate their work in a special logical formalism, namely Description Logics, to allow computers to understand the actual knowledge contained therein. Based on these annotations, computer agents reason about the mathematical knowledge and provide novel services on the Mathematical Semantic Web, giving mathematicians efficient access to vast repositories of mathematics.

This thesis first analyzes the utility of Description Logics for formalizing mathematical knowledge. This analysis concludes with the proposal of a Mathematical Semantic Web which is introduced in great detail subsequently. We elaborate on the architecture and individual building blocks before describing the authoring process for the Mathematical Semantic Web. To motivate the value of our proposal, some services operating on the Mathematical Semantic Web are specified. In an effort to improve knowledge retrieval even further, we introduce the combination of domain and structural semantics for reasoning processes, thereby effectively leveraging additional knowledge.

Finally, we provide a list of Best Practices which aim at simplifying the knowledge modeling process and improving the quality of the resulting knowledge base. The Best Practices have been distilled from the experience gained during our case studies.

[1] Homepage at http://www.zblmath.fiz-karlsruhe.de/MATH/home

[2] Homepage at http://www.arxiv.org. Statistics retrieved on August 17th, 2007.

Contents

1 Foundations

1.1 Overview

The section "Foundations" will review the theories and technologies which this work is based upon, emphazising those aspects which are relevant for the following chapters.
First, the Semantic Web is introduced and an overview of the basic architectural building blocks is given. The chapter is followed by an introduction to Description Logics, their syntax and semantics and a list of reasoning services available.
Both chapters do not claim to cover the subject matter exhaustively and familiarity with some fundamental principles of Computer Science is assumed.

1.2 Semantic Web

The Semantic Web can be best understood as an extension to the current World Wide Web with the aim to enable machine agents to understand and act upon information published online [Berners-Lee et al., 2001]. To motivate the Semantic Web let us consider the vast amounts of information accessible on the Internet. This information is made available primarily as HTML documents which contain a mixture of text and so called tags, which explicate the structure of the document and determine its visual appearance. Machines can retrieve and display such documents but the consumption of contained information remains exclusively with the human user. In other words, the computer agent's understanding of arbitrary documents on the Web does not surpass the character level in that the meaning of sentences and paragraphs is inaccessible. This fact greatly limits software agent's interaction possibilities with the enormous pool of documents. For instance information retrieval currently relies on key word search which is in essence matching on character level enriched by statistical evaluations. Todays search engines cannot determine documents which contain the kind of information the user is looking for but only provide a list of suggestions which results in the user having to actively seek for it.
In short, web content lacks machine accessible semantics which severely restricts computer agents in supporting human users interacting with the World Wide Web and the vast sources of information it contains [Antoniou and VanHarmelen, 2004].

The Semantic Web overcomes this limitation by specifying means to give meaning to elements and resources within web documents. In particular it allows to define kinds of things (e.g. movie), specific instances (e.g. Casablanca) and relationships between instances (e.g. the "stars-in" relationship as in: Ingrid Bergman stars in Casablanca). These definitions can be added to web documents in accordance to a standardized formalism which ensures machine readability. Enriching documents with semantic definitions is called annotation and demonstrates the extensional character of the Semantic Web: meaning is given to elements on the

web by explicitly adding it to the document. It is important to note that semantic annotation is dependent on human contribution, since only humans can understand the intrinsic meaning.

1.2.1 Semantic Web Ingredients

The following touches upon the key features of the Semantic Web which are of relevance for understanding its workings and its architecture [Shadbolt et al., 2006]. Figure 1 illustrates the building blocks of the Semantic Web and their relationship in terms of a layered cake diagram.
It shall be noted that this overview is not meant to serve as a self contained introduction to the Semantic Web but rather outlines those aspects that will be of relevance subsequently. Some aspects that do not pertain to this work are consequently not addressed. For an in depth coverage of the Semantic Web the reader is referred to [Antoniou and VanHarmelen, 2004].

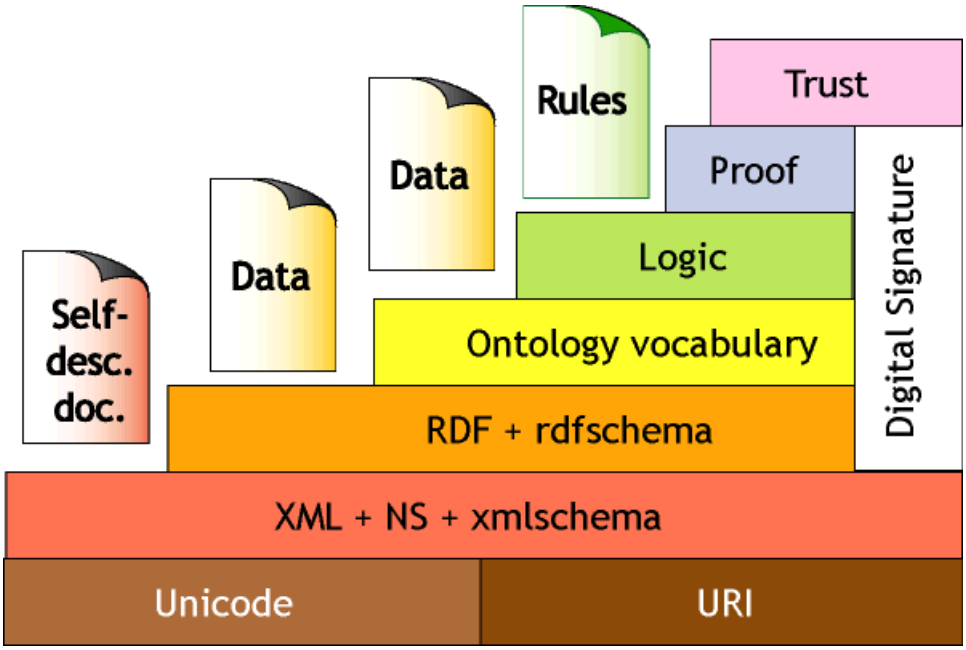

Figure 1: Semantic Web Layered Cake Diagram [Sem,]

Uniform Resource Identifier Uniform Resource Identifiers (URIs) unambiguously identify resources on the web as the name suggests. Using a global naming convention one arrives at a functional mapping between names and resources. Hence it avoids name clashes by design which is a prerequisite for communication over the web. URL (Uniform Resource Locator) is a widely known type of URI.
URIs are a fundamental building block of the Semantic Web in that all elements of

the discourse context need to be assigned a URI prior to being subject to any annotation. It is thereby ensured that the resulting global knowledge base contains no name clashes and that all knowledge objects can be unambiguously identified.

Extensible Markup Language The Extensible Markup Language (XML) is a general purpose markup language and the standard format for exchanging data across information systems. Through its extensible nature, XML can be adopted to virtually any application domain and enjoys extensive tool support. Being a text format based on Unicode XML is furthermore system independent.
For these reasons the Semantic Web relies on XML as the data format for storing knowledge objects.

Resource Description Framework After having identified all knowledge objects in the domain of discourse via URIs, the Resource Description Framework allows to make statements about these objects in terms of triples of the respective URIs. The semantics of the RDF triples is that the first element is the subject, the third element the object and the second element specifies the relation between subject and object. As an example consider (Ingrid-Bergmann,stars-in,Casablanca)[3] which carries the meaning "Ingrid Bergmann stars in Casablanca". Furthermore RDF includes some relationships and objects which carry a predefined meaning, as for instance the "type" relationship or the "Class" object. RDF triples themselves can also be considered as objects which allows for higher-order statements.

Web Ontology Language The Web Ontology Language (OWL) gives RDF greater expressiveness. OWL includes more semantically predefined objects and relationships which allows to assign the knowledge objects a richer meaning. It is for instance possible to define that a relationship is transitive.
OWL was greatly influenced by the theory of Description Logics[4] in that it borrows the constructs and semantics of Description Logics and combines them with the semantics of RDF. One distinguishes between three language variants of OWL: OWL Lite, OWL DL and OWL Full. Each former language is properly contained in the latter. For our purposes, the most interesting of these is OWL DL as this language restricts RDF to an expressive and decidable Description Logic variant (namely $\mathcal{SHOIN}(\mathcal{D})$). Therefore OWL DL inherits the clear semantics and computational properties of Description Logics.

1.2.2 Semantic Web today

Despite steady progress and significant achievements, the original vision of the Semantic Web has not yet materialized nor could one readily assume that it will materialize in the near future. We do not yet see the gigantic knowledge base

[3] these are obviously not URIs, but for the sake of this example we consider them as such.
[4] Description logics is addressed in section 1.3.

created on the web, since most semantic web applications still originate in the research realm.
This shall not be mistaken as an argument for the failure of the Semantic Web. It merely portrays that major achievements have to be made before visions can become reality. Amongst others, the complexity of specifying the meaning of knowledge objects in Semantic Web formalisms is still a big obstacle for many users. It is therefore important to carefully analyse each Semantic Web project to derive realistic expectations.

1.2.3 Semantic Web for Mathematics

Realizing the need of the mathematical community to manage mathematical knowledge should be accepted without argument and justifies the research field of Mathematical Knowledge Management (MKM). In light of this observation it seems natural to consider the possible contributions of the Semantic Web to MKM:

- **Decentralized, heterogeneous knowledge base**
  The Semantic Web provides the technology and infrastructure extension for a scalable, decentralized, heterogeneous knowledge base over the web. Like the Internet, the Semantic Web is decentralized by design and allows all authors to contribute as barriers to publish are low. Using URIs to identify the knowledge objects creates a knowledge base that will be heterogeneous through the lack of central authority, but one that can potentially be integrated.

- **Tool and system support**
  As the emerging standard[5] for knowledge sharing, the Semantic Web does enjoy some tool and system support and will do so even more in the future. Hence building on top of the Semantic Web ensures compatibility and allows to reuse or extend existing tools, thereby joining a large community.

- **Description Logics**
  The Semantic Web and OWL DL in particular are majorly influenced by Description Logics and therefore adopted their clear semantics. Description logics and therefore OWL DL might prove to be a useful logical formalism for mathematics. The utility of Description Logics for formalizing mathematics is carefully analysed in chapter 2.

This initial analysis is continued in subsequent chapters to motivate the proposal of a Mathematical Semantic Web.

[5]The Semantic Web building blocks listed above are each official W3C standards or recommendations

1.3 Description Logic

1.3.1 What is Description Logic?

Description Logics are a family of knowledge representation (KR) formalisms that represent the knowledge of an application domain in terms of its terminology, properties and instances [Baader et al., 2003, chapter 2]. Inspired by earlier KR languages like frame-based systems or semantic networks, Description Logics try to facilitate intuitive knowledge modeling, but unlike their predecessors, they are equipped with a formal logic-based semantics [Baader et al., 2003, chapter 4].
In fact, translations from Description Logics (DL) into First Order Logic (FOL) have been defined [Baader et al., 2003, chapter 2], establishing DL as subset of FOL. A proper subset indeed, in that DL is restricted to a decidable fragment of FOL[6]. Alternatively, DL can be considered as a variant of propositional modal logics.
KR systems based on Description Logics provide facilities to set up knowledge bases, to reason about their content, and to manipulate them [Baader et al., 2003, chapter 2]. Description Logic formalisms define concepts (e.g. movie), relations between concepts (e.g. stars in) and instances of concepts (e.g. Casablanca). DL is therefore by design a candidate formalism for capturing formal ontologies. Formal ontologies serve as a specification of common conceptualizations [Gruber, 1993] which describe the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [Genesereth and Nilsson, 1987]. This fact and the analysis in the following section explain the choice of DL as the underlying KR formalism for the Semantic Web.

1.3.2 Why Description Logic?

To demonstrate the usefulness of yet another logical formalism, some key features of Description Logics are outlined and contrasted to existing KR formalisms.

Computational Complexity In contrast to First Order Logic, Description Logics are decidable. In fact, complexity classes for all languages of the Description Logic family have been determined - ranging from Polynomial to NExpTime-Completenesss [Baader et al., 2003, chapter 3]. On top of these theoretical results, efficient reasoning algorithms have been invented and implemented, yielding powerful reasoning systems which operate on large knowledge bases [Baader et al., 2003, chapter 9].

Clear Semantics The predecessors of Description Logics, most prominently frame-based systems and semantic networks, do not provide a clear semantics

[6]Most DLs can be mapped into the FOL fragments $\mathcal{L}_2$ or $\mathcal{C}_2$, which essentially are restrictions of FOL to two variables.

which means that the semantics is partly encoded in the actual implementation. DL defines a universal, well-defined semantics in terms of FOL or set theory and thereby ensures that knowledge bases formalized in DL are semantically compatible. This is another reason for the active reasoning system development.

Intuitive Modeling The severe limitation of Description Logic's expressiveness compared to First Order Logic is also an advantage, since it greatly reduces the complexity of syntax and semantics for the human knowledge engineer. For instance, knowledge modeling in DL goes without free and bound variables or issues like variable scoping. In fact, the syntax of DL (see 1.3.3) deviates significantly from FOL in that it centers around concept definitions, relations and instances. This facilitates an intuitive approach to knowledge modeling because the syntax of DL resonates to some degree with the human conceptual understanding of the world. This argument is similar to the one made in favor of object oriented programming[7].

1.3.3 Syntax and Semantics

This section provides a brief overview of the syntax and semantics of Description Logics and does not serve as a thorough treatment of the subject matter, but rather introduces the notation and an understanding of the semantics on which subsequent chapters are based. Furthermore the semantic specification is kept informal and appeals to intuition. For a formal and detailed specification of DL syntax and semantics the reader is referred to [Baader et al., 2003, chapter 2].

A DL knowledge base comprises two components, the TBox and the ABox. The TBox captures the terminology[8], in other words it contains the vocabulary of the knowledge domain, whereas the ABox declares individuals and assertions about these individuals[9] in terms of the vocabulary of the TBox.
The vocabulary of the application domain captured in the TBox consists of concepts, which are classes of individuals[10], and binary relations between individuals[11]. In line with earlier examples, the TBox could contain concepts such as "movie" or "actor" and relations like "stars in" between the two.

[7]The similarity between the Object Oriented Programming paradigm and the Description Logic formalism are partly obvious: classes resemble concepts, objects resemble instances and attributes are somewhat similar to relations. OOP talks about inheritance which is similar to subsumption in DL. Despite these similarities important differences exist, for instance the different treatment of relations.

[8]The name TBox stand for "Terminological Box".

[9]The name ABox stands for "Assertional Box".

[10]Concepts were called "kinds of Things" in previous chapters

[11]DL binary relations are also frequently called "roles" or "properties" to distinguish them from their mathematical counterpart. In the interest of understanding by mathematicians we do not make this distinction in this paper.

In addition to atomic concepts and relations (these are also called "primitive definitions") which are simply defined by specifying a name and carry only implicit semantics, DL allows to build more complex concept descriptions out of existing ones using concept constructors and to classify declared relations. The concept constructors and relation classifiers available to define concept descriptions determine the particular language of the Description Logic family. In the following one particular DL language, namely $\mathcal{SHIN}$, is introduced, since the remainder of this work is primarily concerned with this DL language[12].

Table 1 lists all concept constructors and relation classifiers of the Description Logic $\mathcal{SHIN}$. The first column contains the syntax, the second an explanation, the third the semantics in terms of set theory and finally the forth column gives an example from mathematics. A and B are atomic concepts, R an atomic relation and C and D are complex concept descriptions which are defined in terms of A,B, and R. $z \in \mathbb{N}$. In addition to the constructs listed it is possible to restrict the domain and range of relations to certain concept descriptions, which carries the obvious semantics.

1.3.4 Reasoning

Facts stored in the knowledge base are axioms - statements that are assumed to be true in the application domain they describe. From these axioms new statements, statements not stored in the knowledge base, can be inferred, thereby making implicit knowledge explicit.
For example, table 2 displays a simple knowledge base about mathematical functions (TBox only).

This knowledge base (KB) contains the atomic concepts *Function*, $\mathcal{C}$*-Function* and *IntegrableFunction*, the atomic role *has-derivative* and the defined concepts $\mathcal{C}^1$*-Function* and *DifferentiableFunction*. The KB in 2 establishes that

- continuous functions are functions,
- differentiable functions are exactly those functions which have a derivative (which is again a function),
- $\mathcal{C}^1$ functions are exactly those functions which have a continuous derivative,
- differentiable functions are continuous,
- continuous functions are (Riemann) integrable.

[12]To be precise, the Mathematical Semantic Web proposed is build upon OWL DL for obvious reasons and OWL DL is equivalent to the Description Logic $\mathcal{SHOIN}(\mathcal{D})$. But since nominals and datatype properties are of minor importance for this work the introduction of $\mathcal{SHIN}$ suffices.

Syntax	Explanation	Semantics	Example
A, B	Atomic concepts	A,B denote sets of individuals	OddNumbers
$A \equiv B$	Concept and relation equivalence	$A = B$	EvenNumbers $\equiv$ NumbersDivisibleByTwo
$A \sqsubseteq B$	Concept and relation subsumption	$A \subseteq B$	EvenNumbers $\sqsubseteq$ Numbers
R	Atomic relation	R denotes a binary relation	has-domain
$\bot$	Bottom concept / nothing	$\emptyset$	-
$\top$	Top concept / everything	$\bar{\emptyset}$	-
$A \sqcap B$	Intersection of concepts	$A \cap B$	bijectiveFunction $\equiv$ InjectiveFunction $\sqcap$ SurjectiveFunction
$A \sqcup B$	Union of concepts	$A \cup B$	Numbers $\equiv$ OddNumbers $\sqcup$ EvenNumbers
$\neg A$	Complement of concept	$\bar{A}$	OddNumbers $\equiv \neg$ EvenNumbers $\sqcap$ Numbers
$\exists R.B$	existensial quantification	$\{a \vert \exists b.(a,b) \in R\}$	RealFunction $\sqsubseteq \exists$ has-domain.RealNumbers
$\forall R.B$	value restriction	$\{a \vert \forall b.(a,b) \in R :b \in Bt\}$	Relation $\sqsubseteq \forall$ has-domain.Set
$\leq_z R$	maximum cardinality	$\{a \vert$ Let$B = \{b \vert (a,b) \in R\}$ then $\#B \leq z\}$	binaryRelation $\sqsubseteq \leq_2$ has-domain
$\geq_z R$	minimum cardinality	$\{a \vert$ Let$B = \{b \vert (a,b) \in R\}$ then $\#B \geq z\}$	binaryRelation $\sqsubseteq \geq_2$ has-domain
R^{-1}	inverse relation	$\{(a,b) \vert (b,a) \in R\}$	is-domain-of^{-1} $\equiv$ has-domain
Function(R)	R is a functional relation	R is a function	inverse-of
Transitive(R)	R is a transitive relation	R is transitive	divisible-by
Symmetric(R)	R is a symmetric relation	R is symmetric	equal-to

Table 1: DL constructs

$\mathcal{C}$-Function	$\sqsubseteq$	Function
DifferentiableFunction	$\equiv$	Function $\sqcap \exists$ has-derivative.Function
$\mathcal{C}^1$-Function	$\equiv$	Function $\sqcap \exists$ has-derivative.$\mathcal{C}$-Function
DifferentiableFunction	$\sqsubseteq$	$\mathcal{C}$-Function
$\mathcal{C}$-Function	$\sqsubseteq$	IntegrableFunction

Table 2: Simple Knowledge Base about Functions

Even though this KB is very small and simple it already contains some interesting facts about mathematical functions. Furthermore it contains some implicit facts such as for instance:

$$\mathcal{C}^1\text{-}\mathit{Function} \sqsubseteq \mathit{IntegrableFunction}$$

Namely, that continuously differentiable functions are integrable. Although this is a trivial conclusion from the axioms given and bluntly obvious for any mathematician, the inference of this implicit knowledge already requires 3 inference steps[13].

Description Logics come with sound and complete inference algorithms implemented in highly optimized reasoning systems which allow for efficient discovery of implicit knowledge in the KB. The following types of reasoning are supported by Description Logics and commonly provided by DL reasoning systems [Baader et al., 2003, chapter 2], [Sattler,]:

Reasoning services for TBox:

- **Satisfiability:**
  Checks whether a specified concept description C is satisfiable, that is, whether there could possibly exist a world in which all KB axioms hold true and C contains instances.

- **Subsumption:**
  Checks for two specified concept descriptions C, D whether it follows from the KB that C is subsumed by D, that is, whether $C \sqsubseteq D$. **Equivalence** checks whether $C \sqsubseteq D$ and $D \sqsubseteq C$ which is obviously identical to $C \equiv D$. **Disjointness** attempts to proof $C \sqcap D \sqsubseteq \bot$ and returns positively on success.

- **Consistency:**
  A consistency test verifies that statements stored in the KB are logically consistent which means that the KB does not contain any contradictions. As statements are added incrementally to the KB, consistencies can (accidentally) arise. The knowledge base 3 exemplifies an inconsistency:
  Consistency tests can run over the entire KB (including the ABox) by

[13]Continuously differentiable functions are differentiable functions which are continuous functions which are integrable.

A	$\sqsubseteq$	$\exists R.E$
A	$\sqsubseteq$	$\forall R.F$
$E \sqcap F$	$\sqsubseteq$	$\bot$

Table 3: Inconsistent Knowledge Base about pseudo concepts

verifying that the ABox is consistent with respect to the TBox.

- **Classification:**
  A classification service returns the entire subsumption graph (taxonomy) for a given knowledge base.

Reasoning services for ABox:

- **Instance test**
  For a given instance I and concept C it is checked whether I is an instance of C ($I \in C$).

- **Realization**
  Realization computes all instance relations between ABox objects and TBox concepts, that is, for every instance I all concepts C are computed for which it holds that $I \in C$.

- **Instance retrieval**
  For a given concept C all instances I are retrieved from the knowledge base ABox such that $I \in C$.

Going into the details of DL inference algorithms is unfortunately beyond the scope of this brief introduction. The following chapters will refer to these reasoning services for particular examples, but an understanding of the underlying machinery is not necessary.

2 Formalizing Mathematics in DL

2.1 Overview

Having reviewed Description Logics as a modern logic formalism for capturing knowledge, the question of whether Description Logics is a potential language for formalizing mathematical knowledge naturally arises. The following provides an answer to this questions based on a thorough analysis.

2.2 Formalizing Mathematics

Formalizing mathematics refers to the act of translating mathematical documents into an unambiguous, machine readable formalism. In addition to the translation into a rigorous representation, formalization is concerned with making implicit mathematical knowledge explicit. For instance, a written proof often includes informal steps, appeals to intuition or states that something is "trivial". The gaps are automatically filled in by the mathematically inclined mind but need to be explicitly elicitated for a formal representation.

The major driving force behind the formalization of mathematics is automated theorem proving (and proof verification which is closely related). The research field of automated theorem proving employs computer programs to prove mathematical theorems with little to no human intervention. Theorem provers feed on formalized mathematics and also contribute the proofs they generate.

With an increasing interest in automated theorem proving the body of formalized mathematics has continued to increase, with results spanning all major branches of mathematics [Aboul-Hosn, 2006].
The Mizar Mathematical Library (MML) is one of the most prominent libraries of formalized mathematics [Aboul-Hosn, 2006].[14] Documents contained in the MML are called articles and are written in the MIZAR language. Articles contain a set of related theorems and references to other articles they are based on.

In assessing the utility of Description Logics as a formalism for capturing mathematical knowledge one has to compare it to existing mathematical knowledge repositories which are primarily of the form of formalized libraries as outlined above. With the Mizar Mathematical Library being one such formalized repository, the MML and the MIZAR language on which it is based are used as representative examples for the arguments made in the following.

[14] The author is aware of other mathematical formalization languages but believes that they share those properties which are relevant for the analysis of this chapter. Hence the MIZAR project was chosen as an illustrative representative.

2.3 Benefits of Description Logics

Let us first consider the benefits of using Description Logics as a formalism for capturing mathematics which are mainly twofold: decidability and simplicity of knowledge modeling.

2.3.1 Computational Complexity

Description Logics enjoy favorable computational properties compared to other logical formalisms as outlined in chapter 1.3.2. Libraries of formalized mathematics, and the MML in particular, are based on languages which are at least as expressive as First Order Logics and therefore suffer from being undecidable as well. Despite the continuous improvement of automated theorem provers, through the development of novel strategies and advancements in heuristic design [Aboul-Hosn, 2006], the intrinsic complexity of First and Higher Order Logic cannot be overcome.
Due to its computational advantages, Description Logics can hence support much larger mathematical knowledge bases and provide efficient reasoning systems operating on those.
To conclude, the global, all-encompassing mathematical knowledge base which we propose in chapter 3 can only be achieved with Description Logics at its core.

2.3.2 Ease of Knowledge Modeling

In section 1.3.2 it was argued that Description Logics facilitate an intuitive approach to knowledge modeling and thereby simplifies the modeling process. To extend this argument, consider the randomly chosen excerpt from the Mizar Mathematical Library shown in the following listing[15].

```
theorem Th20:
  y in Sphere(x,r) implies LSeg(x,y) \ {x,y} c= Ball(x,r)
  proof
    assume
A1: y in Sphere(x,r);
    per cases;
    suppose
A2:   r = 0;
      reconsider xe = x as Point of Euclid n by TOPREAL3:13;
      Sphere(x,r) = Sphere(xe,r) by TOPREAL9:15;
      then Sphere(x,r) = {x} by A2,TOPREAL6:62;
      then
A3:   x = y by A1,TARSKI:def 1;
A4:   LSeg(x,x) = {x} by TOPREAL1:7;
A5:   {x,x} = {x} by ENUMSET1:69;
      {x} \ {x} = {} by XBOOLE_1:37;
```

[15]This example was retrieved from: ftp://mizar.uwb.edu.pl/pub/version/mml/jordan.miz on Mai 3rd, 2007

```
    hence thesis by A4,A3,A5,XBOOLE_1:2;
  end;
```

Listing 1: An example from the MIZAR library

Without going into the details of what this extract actually means, it should be readily understood that formalizing mathematics in the MIZAR language is not intuitive and requires a major time investment into understanding the MIZAR language first.
The syntactic and semantic complexity is certainly not a characteristic of the MIZAR language alone but a general phenomenon. Since current formalization languages require the expressiveness of First or even Higher Order Logics they have to expose their users to free and bound variables, richer syntax, scoping and all the intricacies that arise from these. In abstract terms, we argue that the complexity of a logical formalism in terms of human understanding comes as a trade off of expressiveness.[16]

On the other hand, the discourse realm of Description Logics is limited to concepts, relations and instances and the syntax is restricted to those constructs listed in table 1. More importantly, the concept constructors of DL resemble those of naive set theory and the semantics can be given in concise set-theoretic statements. Therefore understanding knowledge modeling in Description Logics requires little more than understanding basic set theory, which, without loss of generality, we assume to hold for all mathematicians.
In addition to having less complexity on the statement level, knowledge modeling in DL is very flexible. Concepts can be introduced as atoms first and statements added later to the KB can constrain or even define the original concept. Similarly, properties of relations can be added incrementally. Hence, knowledge bases in DL are easily extensible and allow for collaboration in the knowledge acquisition process.

Becoming proficient with knowledge modeling in DL of course needs a closer look into the subject and experience, but we argue that DL offers a much steeper learning curve compared to other formalisms and hence a significantly lower entry barrier. Consequently we argue that a significant fraction of the mathematical community could actively contribute to a global knowledge base based on DL once they are given a large enough incentive to do so[17]. This is a prerequisite for a vibrant Mathematical Semantic Web as proposed in chapter 3 and in sharp contrast to existing libraries of formalized mathematics which are extended primarily by

[16] Authors have argued [Fürst et al., 2003] that the complexity of First Order Logic can be reduced through visual representations which for instance gave rise to the theory of Conceptual Graphs. We argue that complexity is inherent in expressiveness and therefore the benefit of visual representations seems questionable. In particular, lengthy FOL formulas are easier to comprehend in their concise syntactic form than as an overwhelming graphs.

[17] Chapter 3 elaborates on possible incentives.

a small community of logicians that translate mathematical documents. Rarely does the author of a mathematical paper himself provide a machine readable formalization of his work, which, we argue, could change through the Mathematical Semantic Web.

2.4 Disadvantages of Description Logics

The main disadvantage of Description Logics compared to other logical formalisms is its limited expressiveness. DL can be identified with a fragment of First Order Logics limited to two variables (a so called "guarded fragment" [Andréka et al., 1996]). It is a trivial conclusion that therefore only a fragment of what can be expressed in FOL or even HOL can actually be captured in DL.

The following considers some examples to gain an intuition about how this limitation shapes knowledge modeling in practice.
First, we consider differentiable functions and give a definition which is found in many mathematical textbooks.

Definition: A function $f : \mathbb{R} \to \mathbb{R}$ is differentiable on $\mathbb{R}$ if $\forall a \in \mathbb{R}$ the limit $\lim_{x \to a} \frac{f(x)-f(a)}{x-a}$ exists.

This definition cannot be captured in Description Logics since it contains two scoped variables a and x. DL does not allow the introduction of variables and provides no scoping constructors. To truly understand the limitations of Description Logics we have to abstract from the actual definition given above. As repeatedly stated before, DL defines concepts, relations and instances. It is important to understand that DL treats these as being disjoint. In the example definition above we would like to define the concept of a differentiable function on $\mathbb{R}$, hence we would like to introduce the following concept to our DL knowledge base:

$$\mathit{DifferentiableFunction}\mathbb{R} \sqsubseteq \top \tag{1}$$

But in defining what it means for a function to be differentiable, we have to analyse limits of each individual function. Hence for defining the concept DifferentiableFunction$\mathbb{R}$ we would have to explicitly refer to instances of this concept and treat them as objects in their own right which is not possible in DL. DL only allows to specify how instances of a certain concept relate to other concepts. Hence DifferentiableFunction$\mathbb{R}$ must be declared as an atomic concept and the definition above must remain implicitly captured in the name alone. Of course one could argue for the following definition:

$$\mathit{DifferentiableFunction}\mathbb{R} \equiv \mathit{Function} \sqcap \exists \mathit{has\text{-}derivative.Function} \tag{2}$$

This statement defines the concept of a differentiable function which seems to be superior to the introduction of an atomic concept, but a closer look reveals that the implicit definition of being differentiable is merely moved into the relation *has-derivative*. Hence a judgment on the suitability of the respective statements depends on the intended meaning. If the author would like to express that every differentiable function on $\mathbb{R}$ has a derivative, then the second statement is to be preferred. In case the author wanted to provide a classification of mathematical functions, the first statement suffices.

We conclude this analysis with the observation that the lack of expressivitiy of DL causes major fragments of mathematical knowledge to be captured implicitly in atomic concepts or relations and hence ultimately in names assigned by the knowledge author.
The definition of a monoid, which is given as a second example, supports this conclusion.

Definition: A monoid is a set M with a binary operation $\circ : M \times M \to M$, satisfying the following axioms[18]:

- Associativity: $\forall a, b, c \in M.(a \circ b) \circ c = a \circ (b \circ c)$
- Identity element: $\exists e \in M.\forall a \in M.a \circ e = e \circ a = a$

For the same reasons as above, the exact meaning of "associativity" and "identity element" cannot be captured in DL as any such definition necessarily needs to refer to an actual operation (like $\circ$ in the definition above). Hence, a possible formalization in DL could be

$$\begin{aligned} \mathit{Monoid} \equiv \quad & \exists \mathit{base\text{-}set}.\mathit{Set} \sqcap \exists \mathit{has\text{-}operation}.(\mathit{binaryOperation} \sqcap \mathit{AssociativeOp} \\ & \sqcap \mathit{IdentityElOp}) \end{aligned}$$

where concepts used on the right hand side are assumed to be atomic.

How this observation reflects in the actual modeling of mathematical knowledge is answered in the best practices chapter 4.3. For now the result is used to compare the utility of a mathematical knowledge base in DL with existing formal mathematical libraries such as the MML. As mentioned above, one of the main motivations for creating formalized, machine readable representations of mathematical knowledge is to create repositories for automated theorem provers and to allow for proof verification. In light of the analysis above we can conclude that

[18]Binary operation is assumed to be defined elsewhere. A binary operation is closed.

mathematical knowledge captured in DL is unsuitable for this purpose. It has been shown that large portions of mathematics cannot be translated into Description Logics without abstracting to atomic concepts and thereby effectively loosing valuable knowledge. It is exactly this knowledge that theorem provers need to derive rigorous proofs. For instance proving that all differentiable functions are continuous is trivial given the respective definitions (the definition of differentiable functions is given above). But once we have translated these definitions to DL similar to above, the exact axiomatic definitions have been lost and DL reasoner cannot derive the necessary inference steps.
Consequently, reasoning systems operating on DL knowledge bases are unable to derive non-trivial mathematical proofs. We say "non-trivial proofs" because DL reasoners are of course able to prove trivial theorems. Taking the knowledge base 2 as an example, reasoning algorithms can conclude that all $\mathcal{C}^1$ functions are integrable which is a trivial theorem considering the knowledge explicitly stated. On a site note, we observe that such "trivial" reasoning can provide surprising new insights on large knowledge bases for which Description Logics is very suitable due to its efficiency in reasoning.
On the other hand all non-trivial proofs require an explicit understanding of the axioms underlying a theory which cannot be achieved in DL.

2.5 Conclusion

The last section concluded with the observation that Description Logic is not expressive enough to facilitate non-trivial automated theorem proving. In light of this result one might question their usefulness for mathematics altogether. But having shown that DL cannot be used directly for automated theorem proving does not entail the disutility of formalizing mathematics in DL per se but merely restricts the application of DL to providing helpful services to the most powerful theorem prover currently know: the human being. The next chapter argues for an application of Description Logics with the sole aim of supporting human math workers[19].

[19]The term "math worker" is used here to refer to all groups of people which work with mathematical concepts such as for instance mathematicians, students and engineers.

3 Mathematical Semantic Web

3.1 Overview

In light of the results from previous chapters we propose a Mathematical Semantic Web build upon the Semantic Web and in particular OWL DL, the Description Logic variant recommended by W3C. In the spirit of the Semantic Web, the Mathematical Semantic Web envisions the creation of a vast, decentralized mathematical ontology to which authors of mathematical documents can easily contribute due to the intuitive knowledge modeling paradigm potentiated by DL and the technologies underlying the Semantic Web. Description Logic reasoning systems can operate on this enormous knowledge base and provide valuable services to all types of math workers.

The reasons for developing ontologies on the Mathematical Semantic Web are manifold:

- Novel services build on efficient reasoning systems can feed on the ontologies and support math workers in their daily business. Some possible services are specified in section 3.6.
- Ontologies enable efficient reuse of previously defined knowledge from a wide variety of sources.
- Ontologies capture a common understanding of the conceptualization of the mathematical realm which facilitates information retrieval and agent communication (see chapter 4.4.1).

This chapter first introduces an architectural extension of the Semantic Web by which we arrive at the Mathematical Semantic Web followed by an outline of possible extensions to mathematical authoring environments in order to ease publishing on the Mathematical Semantic Web. After describing the authoring process in its entirety, a careful analysis of the different types of knowledge contained in a document is given and concluded with an application to our work. The chapter ends with the specification of some services that could be realized on the Mathematical Semantic Web. These services provide the necessary incentive for mathematicians to semantically annotate their documents and thereby contribute to a mathematical ontology.

3.2 Ontology Format

On the Semantic Web ontologies are formalized in OWL DL which has an XML based syntax and hence Semantic Web ontologies are in essence XML files. Consequently, OWL DL ontologies are independent files linked through the ontologies they import but without reference to the content that has motivated the ontological definitions. The concepts, relations and instances that are defined in any ontology do not exist in a vacuum. There exists some document, some piece of information or even some thought that has brought about any such definition.

We believe it is of great advantage to add ontological definitions to the documents from which they originate instead of publishing them in independent files. We therefore propose an infrastructure which supports appending formalized knowledge directly to the statement from which it arose. This proposal rests on the fact that humans rarely use a completely formal language for communicating ideas and knowledge and that consequently most formal representations of knowledge are a translation of their counterparts written for human consumption.

Benefits of this approach are manifold:

- **Authoring:**
  Formalizing mathematical statements can be integrated into the authoring process for mathematical documents and does not need to take place in a separate step, thereby decreasing the barrier to formalized knowledge creation. Hence knowledge engineering reduces to a small translation effort which can occur right after the formulation of the original statement.

- **Consistency:**
  Having the formalized variant in close proximity to the original statement eases consistency checking and reduces mistakes made during the translation into Description Logics.

- **User friendliness:**
  For each axiom in the knowledge base it is possible to determine which document, and which statement in particular, motivated its creation. It has been shown in chapter 2.4 that concepts in Description Logics can carry a significant amount of implicit knowledge due to the lack of expressiveness of DL. Hence it is particularly important that users can retrieve the original definition in mathematical vernacular for any given concept (e.g. What does the atomic concept “AssociativeOperation” actually stand for?).

For an example let us reintroduce the ontological definition of a monoid from chapter 2.4:

$$Monoid \equiv \exists base\text{-}set.Set \sqcap \exists has\text{-}operation.(binaryOperation \sqcap AssociativeOp \sqcap IdentityElOp)$$

As in the original example this statement is justified by the corresponding definition and hence we propose appending the formal Description Logic statement to the definition given in mathematical vernacular[20].

An ontological format for mathematics must consequently support structured mathematical vernacular intertwined with formal ontological definitions. The Open Mathematical Documents (OMDoc) [Kohlhase, 2006] has been specifically designed to accommodate this requirement making it the optimal format for our purposes.

OMDoc is a content-oriented markup format for mathematical knowledge and documents. The format represents mathematical knowledge at three levels: *objects*, i.e. content representations of mathematical formulae, *statements*, e.g. definitions, theorems, and proofs, and *theories*, i.e. a structured representation of mathematical context [Kohlhase and Rabe,]. OMDoc is based on XML and includes OpenMath for representing mathematical formulae. The OpenMath format defines an XML encoding of an abstract mathematical object model thereby allowing the meaning of a mathematical object to be encoded and exchanged between systems. The meaning of mathematical objects is given through symbols which are defined in OpenMath content dictionaries or theories in OMDoc.

The next section specifies an embedding of Description Logics into OMDoc via OpenMath and defines all concept constructs available.

3.2.1 OMDoc Ontology Extension

In OMDoc theories are treated as first-class objects that can be structured like documents [Kohlhase and Rabe,]. Theories constitute content dictionaries and contain `symbol` elements declaring a symbol for a mathematical concept, such as $\mathbb{N}$ for the natural numbers, 1 for the natural number "one", and `equality` for the equality relation. [Kohlhase, 2006, chapter 15] states that "we will refer to the mathematical object declared by a `symbol` element as a 'symbol', iff it is usually communicated by specialized notation in mathematical practice, and as a 'concept' otherwise. For our work the second reading is the important one, since notational declarations are of little importance for knowledge representation. The proposed OMDoc ontology extension uses the `symbol` element to declare the

[20]The language used by working mathematicians to communicate is called mathematical vernacular to distinguish it from fully formalized mathematical languages.

concepts, relations and instances used throughout the ontology. The `type` child of the `symbol` element is used to distinguish between the three:

- **Declaration of a concept**
  ```
  <type system="OntLang">
  <om:OMOBJ><om:OMS cd="OntLang" name="concept" /></om:OMOBJ>
  </type>
  ```
- **Declaration of a relation**
  ```
  <type system="OntLang">
  <om:OMOBJ><om:OMS cd="OntLang" name="relation" /></om:OMOBJ>
  </type>
  ```
- **Declaration of an instance**
  ```
  <type system="OntLang">
  <om:OMOBJ><om:OMS cd="OntLang" name="instance" /></om:OMOBJ>
  </type>
  ```

`OntLang` stands for "Ontology Language" and is the name we gave to the content dictionary which contains the description logic constructors to build ontologies in OMDoc as well as other symbols necessary for the ontology extension.

Using the `symbol` element for declaring the ontology vocabulary is a natural choice to make since it is in accordance with the intended usage. The following example declares the concept `Group`, the relation `equality`, and the instance `KochSnowflake`.

```
<symbol name="Group">
  <type system="OntLang">
    <om:OMOBJ><om:OMS cd="OntLang" name="concept" /></om:OMOBJ>
  </type>
</symbol>

<symbol name="equality">
  <type system="OntLang">
    <om:OMOBJ><om:OMS cd="OntLang" name="relation" /></om:OMOBJ>
  </type>
</symbol>

<symbol name="KochSnowflake">
  <type system="OntLang">
    <om:OMOBJ><om:OMS cd="OntLang" name="instance" /></om:OMOBJ>
  </type>
</symbol>
```

Listing 2: Code example using the Ontology Language

OMDoc is a format for structuring mathematical documents by providing elements to mark up definitions, assertions, proofs, etc. as such. These elements

usually have one CMP child and an arbitrary number of FMP children. CMP stands for "Commented Mathematical Property" and contains mathematical vernacular, meaning a mixture of text and formulae. FMP on the other hand stands for "Formal Mathematical Properties" and contains a rigorously formalized equivalent of the contents of the respective CMP element. The one CMP element contains mathematical content geared towards human consumption whereas the FMP elements exhibit the same content represented in different formalisms.
This infrastructure fulfills the requirements towards an ontological format as postulated above in that the ontological definitions can be written into an FMP element which is paired with the CMP element that contains the original statement in mathematical vernacular.
The following snippet illustrates the embedding in OMDoc:

```
<definition xml:id="some.def" for="#SomeConcept" type="simple">
  <CMP>
    This is a definition ...
  </CMP>

  <FMP logic="dl">
    <!-- This is the Description Logic equivalent -->
  </FMP>
</definition>
```

Listing 3: Embedding the Ontology Language in OMDoc

Ontological definitions are written in OpenMath in that we provide a Description Logic in OpenMath syntax by defining concept constructors as OpenMath symbols in the content dictionary "OntLang". This choice was motivated by the fact that OMDoc natively supports OpenMath and tool support exists. Table 4 lists all concept constructs (i.e. OpenMath symbols) and specifies their meaning in terms of the Description Logic introduced in table 1[21]. Table 5 lists instance constructs which can be used to define instances in the ABox.

[21]The "O" in front of all symbol names was included as a global namespace within S.

Concept Constructor	**DL Equivalent**	**Comment**
OTopConcept	$\top$	
OBottomConcept	$\bot$	
OConceptDef(A,B)	$A \equiv B$	Equivalence for concepts
OConceptNecessaryDef(A,B)	$A \sqsubseteq B$	Subsumption for concepts
OConceptEquiv(A,B)	$A \equiv B$	Equivalence for concepts
OConceptSubsume(A,B)	$A \sqsubseteq B$	Subsumption for concepts
OUnion(A,B)	$A \sqcup B$	
OIntersection(A,B)	$A \sqcap B$	
OComplement(A)	$\neg A$	
ODisjoint(A,B)	$A \sqcap B \equiv \bot$	
ORelationDef(R,T)	$R \equiv T$	Equivalence for relations
ORelationNecessaryDef(R,T)	$R \sqsubseteq T$	Subsumption for relation
ORelationEquiv(R,T)	$R \equiv T$	Equivalence for relation
ORelationSubsume(R,T)	$R \sqsubseteq T$	Subsumption for relation
ODomain(R,A)	$\mathbb{D}(R) = A$	Restricts the domain of R to concept description A
ORange(R,A)	$\mathbb{R}(R) = A$	Restricts the range of R to concept description A
OInverse(R,T)	$R^{-1} \equiv T$	
OFunction(R)	Function(R)	
OInverseFunction(R)	Function(R^{-1})	
OTransitiveRel(R)	Transitive(R)	
OSymmetricRel(R)	Symmetric(R)	
ORestrictRelationTo(R,A)	$\forall R.A$	
OExistsRelationWith(R,A)	$\exists R.A$	
OMaxCardinality(R,z)	$\leq_z R$	
OMinCardinality(R,z)	$\geq_z R$	
OExcatCardinality(R,z)	$\leq_z R \sqcap \geq_z R$	

Table 4: Concept constructs

Using the symbols from table 4 one writes ontological definitions as standard OpenMath formulas. The listing below gives an example for the Description Logic statement[22]

$$\mathit{DifferentiableFunction} \equiv \mathit{Function} \sqcap \exists \mathit{has\text{-}derivative}.\mathit{Function} \tag{3}$$

and two instance definitions.

[22]The atomic concept *Function* and the relation *has-derivative* are assumed to be defined elsewhere. *Functions* is the name of the current theory and hence the name of the content dictionary containing the declared symbols. "om" refers to the OpenMath namespace.

Instance Constructor	Type	Explanation
OInstanceOf(I,A)	Definition	Defines an instance I to be of concept A
OInstanceDef(I,A,L)	Definition	Defines an instance I to be of concept A. In addition all properties in the list L have the implicit first argument I. Properties can only occur as the third argument to this construct, in other words, they must be bound.
OAnonymInstanceOf(A)	Definition	Defines an anonymous instance of concept A
OInstanceDef(A,L)	Definition	Defines an anonymous of concept A. In addition all properties in the list L have the anonymous instance as a first argument. Properties can only occur as the third argument to this construct, in other words, they must be bound.
ORelatedTo(R,J)	Property	Defines that $R(I, J)$ holds where I is the implicit first argument, R a relation and J another instance.
OIdenticalTo(J)	Property	$I = J$ where I is the implicit first argument and J another instance
ODifferentFrom(J)	Property	$I \neq J$ where I is the implicit first argument and J another instance

Table 5: Instance constructs

```
<symbol name="DifferentiableFunction">
  <type system="OntLang">
    <om:OMOBJ><om:OMS cd="OntLang" name="concept" /></om:OMOBJ>
  </type>
</symbol>


<FMP logic="dl">
    <om:OMOBJ>
      <om:OMA>
        <om:OMS cd="OntLang" name="OConceptDef" />
        <om:OMS cd="Functions" name="DifferentiableFunction" />
        <om:OMA>
          <om:OMS cd="OntLang" name="OIntersection" />
          <om:OMS cd="Functions" name="Function" />
          <om:OMA>
            <om:OMS cd="OntLang" name="OExistsRelationWith" />
            <om:OMS cd="Functions" name="has-derivative" />
            <om:OMS cd="Functions" name="Function" />
          </om:OMA>
```

```
            </om:OMA>
          </om:OMA>
        </om:OMOBJ>
    </FMP>

    <symbol name="particularF">
      <type system="OntLang">
        <om:OMOBJ><om:OMS cd="OntLang" name="instance" /></om:OMOBJ>
      </type>
    </symbol>
    <symbol name="particularF'">
      <type system="OntLang">
        <om:OMOBJ><om:OMS cd="OntLang" name="instance" /></om:OMOBJ>
      </type>
    </symbol>

    <FMP logic="dl">
        <om:OMOBJ>
          <om:OMA>
            <om:OMS cd="OntLang" name="OInstanceOf" />
            <om:OMS cd="Functions" name="particularF'" />
            <om:OMS cd="Functions" name="Function" />
          </om:OMA>
        </om:OMOBJ>
        <om:OMOBJ>
          <om:OMA>
            <om:OMS cd="OntLang" name="OInstanceDef" />
            <om:OMS cd="Functions" name="particularF" />
            <om:OMS cd="Functions" name="DifferentiableFunction" />
            <om:OMA>
              <om:OMS cd="OntLang" name="ORelatedTo" />
              <om:OMS cd="Functions" name="has-derivative" />
              <om:OMS cd="Functions" name="particularF" />
            </om:OMA>
          </om:OMA>
        </om:OMOBJ>
    </FMP>
```

Listing 4: Full example of a Ontology Language statement embedded in OMDoc

The Description Logic just specified in the syntax of OpenMath is equivalent to $\mathcal{SHIN}$ and hence can be translated to OWL DL via XSLT stylesheets. The translation process is treated in more detail in chapter 3.5.

3.2.2 Design Choices

This section explains some of the choices made when designing the embedded Description Logic language specified in the proceeding section.

Names With mathematicians being the clearly defined target group for the specified DL language, names used in the language, in particular those for concept and

instance constructs, were chosen to sound familiar to users with a mathematical background and to have a similar meaning in mathematics. For example, what has been called "relation" throughout this paper is commonly referred to as "role"in the literature on Description Logics and called "property" in OWL.

Redundant Concept constructs Table 4 seemingly contains redundant concept constructs. *OConceptDef* and *OConceptEquiv* have the same meaning in DL, the same holds true for *OConceptNecessaryDef* and *OConceptSubsume*. In fact those constructs are mapped to the same OWL DL elements during translation. The introduction of two redundant concept constructs was motivated by an observation of mathematical practice. In mathematics concepts are usually defined when they are first being introduced and afterwards assertions about these concepts are being made. For example, let us consider a document treating differentiable functions. First, differentiable functions are defined as a particular type of function. Lacking the expressiveness, one would define differentiable functions as an atomic concept in DL:

$$\mathit{DifferentiableFunction} \sqsubseteq \mathit{Function} \tag{4}$$

Later in the same document the author might assert that differentiable functions are continuous functions which is translated to DL as:

$$\mathit{DifferentiableFunction} \sqsubseteq \mathcal{C}\text{-}\mathit{Function} \tag{5}$$

As one can see, DL does not differentiate between definitions and assertions but treats both as axioms in the knowledge base. For the user on the other hand it can be important to distinguish between the two[23]. For this reason we introduced constructs which have the same meaning in DL. Those constructs ending in "Def" shall be used when defining concepts while the other two constructs are used for capturing assertions.

Anonymous Instances As specified in table 5 the proposed Description Logic language allows the usage of anonymous instances. Anonymous instances are necessary in those cases where one would like to define an instance of interest to be related to another instance without having to declare the second one. For example, if one would like to state that a specific function "Fun" has a derivative (and the derivative itself is irrelevant in the document), an anonymous instance needs to be used which is shown in the listing:

```
<symbol name="Fun">
  <type system="OntLang">
3   <om:OMOBJ><om:OMS cd="OntLang" name="instance" /></om:OMOBJ>
  </type>
```

[23]Chapter 3.5 explains in more detail why there is a need to distinguish and how users can benefit from it. It also outlines the technical implementation of this differentiation.

```
</symbol>

<example xml:id="some.example" for="..." type="for">
  <CMP> ... </CMP>
  <FMP logic="dl">
    <om:OMOBJ>
    <om:OMA>
      <om:OMS cd="OntLang" name="OInstanceDef" />
        <om:OMS cd="Functions" name="particularF" />
        <om:OMS cd="Functions" name="Function" />
        <om:OMA>
          <om:OMS cd="OntLang" name="ORelatedTo" />
          <om:OMS cd="Functions" name="has-derivative" />
          <om:OMA>
            <om:OMS cd="OntLang" name="OAnonymInstanceOf" />
            <om:OMS cd="Functions" name="Function" />
          </om:OMA>
        </om:OMA>
    </om:OMA>
    </om:OMOBJ>
  </FMP>
</example>
```

Listing 5: Declaration of anonymous instances

Anonymous instance constructs (*OAnonymInstanceOf* and *OAnonymInstanceDef*) can be inserted wherever an instance is expected as an argument to an instance construct of type "Property" (see table 5).

Datatype Properties Even though datatype properties are part of OWL DL they are not included in the Math ontology language. Datatype properties allow the definition of relations which have a concept as their domain and a datatype (e.g. integer or string) as their range.

In order to keep the complexity of the Math ontology language as low as possible it was decided to exclude datatype properties in the first prototype. We believe that the coexistence of datatype and object properties (normal relations) could cause confusion, for example between the datatype *xsd:integer* and the concept *NaturalNumber* in the concept definition

$$RationalNumber \equiv \exists nominator.(xsd{:}integer) \sqcap \exists denominator.NaturalNumber \quad (6)$$

where nominator is a datatype property and denominator is an object property. Mixing both types of properties obviously leads to a flawed definition.

If field studies suggest that datatype properties are indeed needed for modeling mathematical knowledge they can easily be added. In fact, there are examples in which datatype properties could potentially be useful. For instance in some

application domains it might make sense to define a finite set as[24]:

$$\mathit{FiniteSet} \equiv \mathit{Set} \sqcap \exists \mathit{number\text{-}of\text{-}elements.(xsd{:}integer)} \tag{7}$$

where *number-of-elements* is a datatype property and *xsd:integer* stands for the integer datatype.

Nominals For the same reason as above nominals have not been included even though they are part of OWL DL. If the necessity for nominals in mathematical knowledge modeling should arise they can easily be added.

Qualified Number Restrictions Since the Mathematical Semantic Web we propose should be compatible to the original Semantic Web to be able to reuse its architecture and infrastructure and since OWL DL does not support qualified number restrictions, these have not been considered for the Math Ontology language.

Additional Constraints The Math Ontology language imposes an additional constraint on ontological statements compared to OWL DL in that the first argument of any of four concept constructs *OConceptDef*,*OConceptNecessaryDef*, *OConceptEquiv*, and *OConceptSubsume* expect a concept name (meaning a symbol name for a symbol of type concept) instead of an arbitrary concept description.

It is believed that this constraint does not restrict knowledge modeling in any sever manner but increases the readability of the DL statements.

It has to be noted that neither this nor other constraints are currently being automatically checked. A future implementation will verify compliance through type checking.

[24]Such a definition is useful in scenarios where one would like to define instances of *FiniteSet* by explicitly stating the number of elements they contain (and without having to define an instance of natural numbers).

3.3 Authoring Environment

In chapter 3.2 we presented with OMDoc an ontology format and with the Math Ontology Language a Description Logic language that can be embedded therein. Unfortunately, editing OMDoc and writing OpenMath formulae by hand is rather cumbersome and since only limited authoring platforms for OMDoc exist it is very unlikely that authors of mathematical documents will write plain XML code in order to enjoy the benefits of the Mathematical Semantic Web. Even if tool support for OMDoc existed it is questionable whether mathematicians would leave the authoring environments they are accustomed to.

3.3.1 sTeX

TEX, with its document markup extension LATEX, is the most popular typesetting program among mathematicians. The strength of TEX lies in accurately typesetting complex mathematical formulae. Although statistics do not exist, it can be safely assumed that TEX/LATEX operates at the core of most authoring environments for mathematics.

In order to enable a large group of mathematicians to contribute to the Mathematical Semantic Web from within the authoring environment they have become accustomed to, it is desirable to allow semantic annotations and Description Logic formulae directly in TEX/LATEX which we achieve through the sTEX macro packages. sTEX is a collection of TEX macro packages that allow to markup TEX/LATEX documents semantically without leaving the document format [Kohlhase, 2004]. Without going into the particularities of sTEX it suffices for our purposes to consider it as a syntactic variant of OMDoc, where the former is based on TEX and LATEX and the later on XML and OpenMath. In fact sTEX was developed with the goal of converting mathematical LATEX documents into OMDoc. Amongst other benefits, sTEX provides support for writing rigorous mathematical formulae using TEX standard macro notation which is to be preferred over OpenMath's XML representation especially for authors who are familiar with TEX.

We implemented an extension in sTEX which adds support for the Math Ontology language presented above. A detailed presentation of the Math Ontology language for sTEX is redundant (the reader is referred to tables 4 and 5) , since the only difference between sTEX and OMDoc with regard to formalizing mathematics in DL is the underlying infrastructure: TEX versus XML. Construct names and general workings are identical. The implementation of all Math Ontology Language constructs can be found in the appendix A.1. The listing below shows the same content as the previous OMDoc example in sTEX notation to illustrate the similarities and notational differences.

```
\symdef[type=concept]{DifferentiableFunction}{differentiable
    function}
```

```
  \begin{FMP}[logic=dl]
    $\OConceptDef {\DifferentiableFunction}
                  {\OIntersection{
                      \Function,
                      \OExistsRelationWith{\hasDerivative,\Function}
                  }}$
  \end{FMP}

  \symdef[type=instance]{particularF}{F}
  \symdef[type=instance]{particularFprime}{F'}

␣␣\begin{FMP}[logic=dl]

␣␣␣␣$\OInstanceOf{\particularFprime}{\Function}␣$␣\\
␣␣␣␣$\OInstanceDef{\particularF}{\DifferentiableFunction}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{\ORelatedTo{\hasDerivative}{\particularFprime}}$
␣␣\end{FMP}
```

Listing 6: Code example in sTeX

Despite both formats being semantically equivalent, the example demonstrates that the sTEX syntax is much easier to comprehend and write. Furthermore sTEX also provides support for specifying the presentation of concepts, relations and instances which is important for mathematical authoring since the primary purpose is to create a document for publication. For example, the symbol definition

```
\symdef[type=instance]{particularFprime}{F'}
```

in the example above does not only declare the instance "particularFprime" but also specifies its appearance, namely F'.

3.3.2 Authoring Environment Evaluation

By integrating ontological annotations into the widely used typesetting program TEX we have effectively reduced the entry barrier to the Mathematical Semantic Web, since the additional effort for modeling mathematical knowledge in terms of Description Logic statements is minimalized. Furthermore, the syntax of our DL variant, the Math Ontology Language, in sTEX resembles the familiar notation of TEX. Together with our earlier efforts to design a Description Logic language which can be readily understood by mathematicians with a background in set theory (see 3.2.2), we have achieved a steep learning curve for the Mathematical Semantic Web and thereby increased the likelihood of wide spread support within the mathematical community[25].

[25] One could argue that in order to create documents which can be converted to OMDoc, the author needs to make the structure of the document explicit in sTEX which requires significant effort. But since an explicit document structure is independent from ontological annotations this argument does not pertain to the analysis.

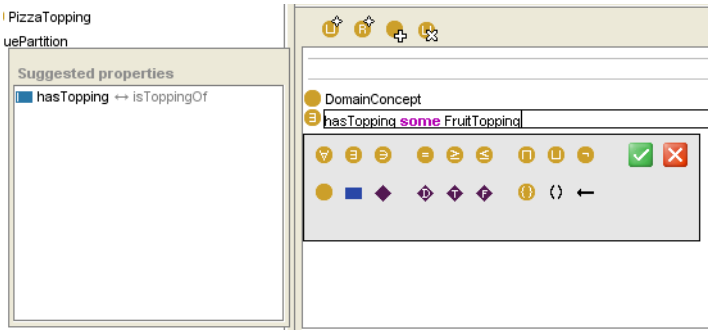

Figure 2: Embedded editor for DL formulas in Protégé

Nevertheless, the usability of this initial prototype is certainly not optimal. The following list suggests improvements and add-ons to the authoring environment which would improve user friendliness and further decrease the effort of crafting formalized mathematical knowledge.

- **DL formula plugin**
  Many editors support LaTeX documents through symbol dictionaries, automatic macro expansion and other useful services. These LaTeX development environments could be extending by a Math Ontology Language plugin, which assists the author in writing DL formulas using the Math Ontology Language constructs. Deeply nested DL formulas in sTeX can become hard to comprehend and write which could be simplified through an embedded editor for DL formulas.

  For example Protégé, which is a general purpose editor for OWL DL and frame-based ontologies, includes such an embedded editor as show in figure 2. [Knublauch et al., 2004] outlines the design choices behind the ProtégéOWL DL plugin and can serve as a starting point for an analysis on a Math Ontology Language editor.

- **Automatic discovery of concepts, relations and instances**
  Especially for authors who are beginning to use the Math Ontology Language to semantically annotate their work it might be helpful if concepts, relations and instances used throughout the document would be automatically detected and presented to the author. Once the author is aware of their existence she will add DL formulas to formalize their meaning.
  Significant research in Artificial Intelligence investigates methods for automatic vocabulary elicitation in the context of the Semantic Web which could also be applied for our purposes with [Witbrock et al., 2004] being one example.

- **Type checking**
  As mentioned in chapter 3.2.2 no type checking is currently in place which

means that it is possible to, for instance, hand a declared instance as a first argument to a concept construct which obviously constitutes a malformed statement. Thus it would be desirable to implement type checking to verify that all DL statement are well-formed, that symbols are not redeclared, and that the additional constraints of the Math Ontology Language are adhered to.

Type checking would alert the author to mistakes made in the knowledge modeling process and thereby improve the quality of the global knowledge base on the Mathematical Semantic Web.

- **Ontology Discovery**
  Most importantly, the author must be supported in discovering existing ontologies which define the concepts, relations, or instances that form the theoretical background of the author's current work. For example, an author writing a paper about differentiable manifolds would greatly benefit from reusing existing ontologies on manifolds instead of having to define all concepts needed for knowledge modeling by herself. The obvious reason for facilitating ontology reuse is that it saves time and effort. More importantly though, ontology reuse is a fundamental prerequisite for the success of the Mathematical Semantic Web.

  To support this claim, let us consider the scenario in which no ontology reuse occurs. Ignoring the significant extra effort for the sake of this argument, authors would predefine all concepts, relations and instances on which their work depends prior to defining newly introduced knowledge objects in terms of the predefined ones. Consequently, the created ontology is independent from any other published mathematical ontology which means that the Mathematical Semantic Web degenerates to a pool of unconnected knowledge "islands".
  Reusing ontologies, on the other hand, creates a network of mathematical ontologies which emerge to the enormous knowledge base that the Mathematical Semantic Web envisions.

  Finding relevant, existing ontologies on the Internet is a non-trivial task due to its size alone. Hence tools must be developed which help the author in discovering those knowledge repositories on which his work depends and automatically import selected ontologies.

  Ontology reuse is identical to theory import in OMDoc [Kohlhase, 2006]. In OMDoc mathematical knowledge is subdivided into theories which contain related knowledge objects or one specific conceptualization (e.g. `Group`).

Theories in OMDoc import other theories on which they depend (e.g. `Group` imports `Mapping` which contains the concept "Function"). In light of this similarity ontology reuse is implemented on top of the theory import infrastructure provided by OMDoc.

An advanced service for ontology discovery would not just return all ontologies which pertain to the author's work but only those which are indeed relevant. Relevance could be determined based on the community of practice of which the author is a member. This idea and its implications are elaborated in chapter 4.4.1.

3.4 Authoring Process

The previous sections have described the building blocks of the Mathematical Semantic Web in detail, hence it is now time to fit the individual pieces together and introduce the entire authoring process for mathematical documents enriched with ontological annotations.
The prototypical authoring process on the Mathematical Semantic Web is visualized in figure 3. The individual steps of the process are numbered and explained in the following.

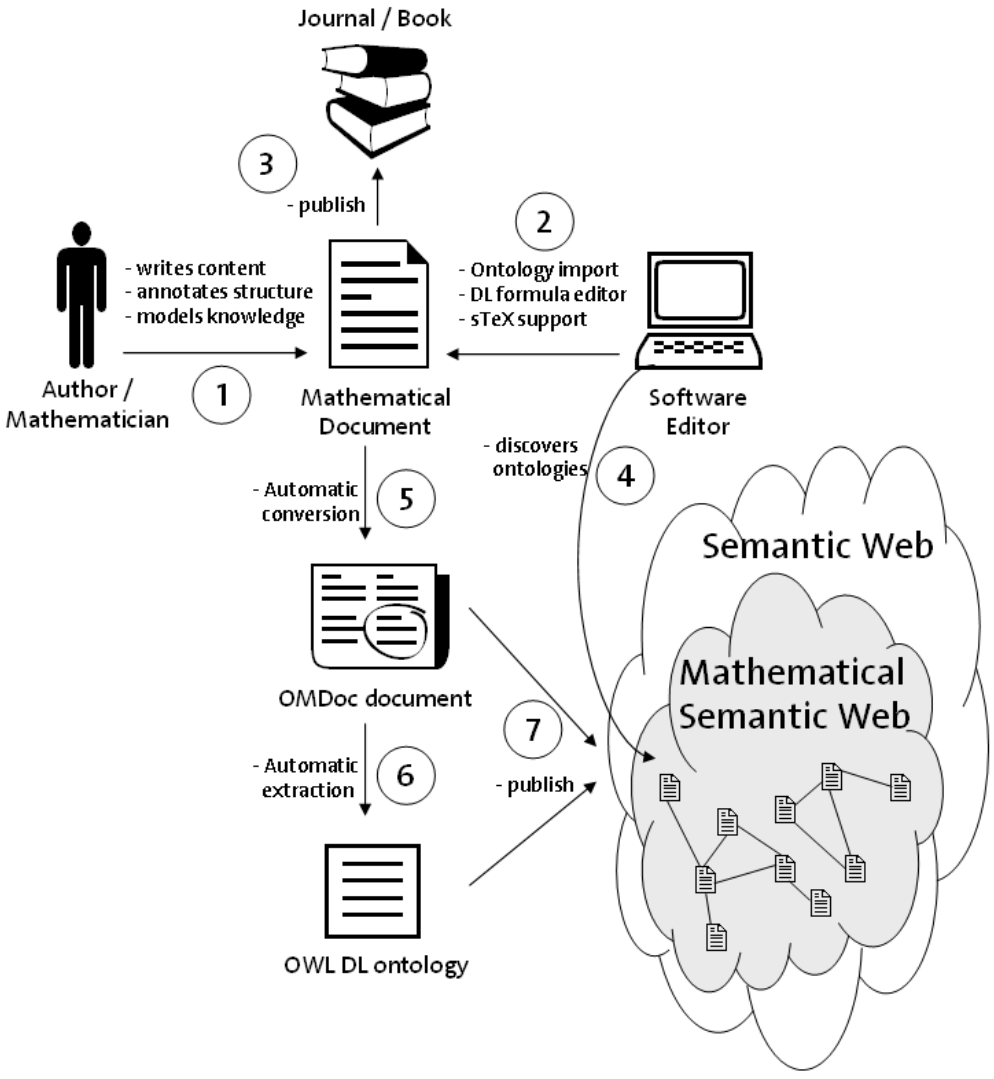

Figure 3: Mathematical Semantic Web Authoring Process

1. The author writes a mathematical document, which includes:

- writing mathematical vernacular in LaTeX and mathematical formulae in sTeX,
- making the structure of the document explicit via sTeX markup,
- modeling the contained mathematical knowledge in the Math Ontology Language.

2. An editor software supports the author in her work by importing necessary ontologies (which are discovered on the internet, see step 4) and providing advanced editing functionalities (see 3.3.2 for more information).

3. The written document is published in a journal, printed as a book, uploaded to the internet or distributed via different channels.

4. The editor software finds relevant ontologies on the Mathematical Semantic Web (see 3.3.2 for details).

5. The sTeX document is automatically converted to OMDoc using the LaTeXML converter developed by Bruce Miller [Miller,] and special sTeX bindings for OMDoc.

6. The ontological statements are automatically extracted from the generated OMDoc document and converted into an OWL DL ontology using XSLT stylesheets. This step is necessary in order for the Mathematical Semantic Web to be compatible with the Semantic Web on which it is based, since the OpenMath based Mathematical Ontology Language embedded in OMDoc is not natively supported by Semantic Web technologies and services. Extracting an OWL DL ontology allows us to benefit from existing and future Semantic Web services. Furthermore, since the OWL DL ontology is extracted fully automatically, our approach does provide the advantages analysed in section 3.2.

7. Both, the generated OMDoc document and the extracted OWL DL ontology are published on the internet and hence become part of the Mathematical Semantic Web. The OMDoc document and the OWL DL ontology are closely related by construction and therefore linked together.

One should note that despite the seemingly complicated nature of the authoring process the author notices only little change, since most of the process steps are executed automatically. The only change apparent to author derives from the fact that she adds ontological statements using the Math Ontology Language to the document which reflects in a slightly modified authoring environment (see chapter 3.3).

3.5 Uniting Structural and Domain Semantics

It is common practice in mathematics to structure documents by declaring definitions, theorems, proofs, etc. as such. Structured mathematical expositions are easier to write, but, more importantly, easier to understand and comprehend. Authors most frequently specify the document structure though keywords (e.g. "definition") and special formatting (e.g. bold face) thereby making the document structure obvious for the human reader but unfortunately inaccessible for computer agents. One of the advantages of using OMDoc is that it provides markup to make the structure of a mathematical document explicit. We call a document structure explicit if it is machine processable.

That the structure of a document helps in understanding its content is undisputed, hence we argue that the document structure in fact carries its own semantics in that it frames the meaning of the content and thereby adds to its meaning (irrespective of the actual content). In line with previous work [Kohlhase, 2006], we call this semantics "structural semantics" as it refers to the meaning associated with the structural elements of a given document structure.
The keyword "Definition" in front of a paragraph, for example, tells the human user that the text to follow will introduce a new mathematical concept by defining it in terms of known ones. This is in contrast to "Theorem" which specifies that the paragraph contains a novel true statement about known concepts.

One can therefore distinguish between two types of knowledge contained in a document: structural and domain knowledge. Domain knowledge is the knowledge represented by the content of a document[26]. Domain knowledge is certainly more important than structural knowledge to both, the reader and the author of a document, since structural knowledge is useless by itself. But structural knowledge can add to a human's understanding of domain knowledge as argued above, hence structural knowledge can enrich domain knowledge.

Unfortunately, knowledge management has so far addressed only domain knowledge and ignored structural knowledge. The following examples illustrate that bringing both types of knowledge together is beneficial in practice:

- **Finding examples**
  A mathematics teacher would like to post a homework question which asks the students to prove that a certain mathematical object is a group. Hence the teacher is interested in retrieving all documents which introduce instances of the concept `group` in an example.
  Finding all instances of the concept `group` requires reasoning on domain

[26] The domain of interest for our research is mathematics and hence our work has only been concerned with mathematical knowledge up until this point. But the argument made here is independent from the application domain.

knowledge, whereas retrieving documents in which these instances occur as an example is accomplished through reasoning on structural knowledge.

- **Finding theorems**
  A mathematician is conducting research on Julia sets and currently does not progress on a particular proof. Hence he would like to retrieve all assertions made about Julia sets and related concepts to find out whether any of these could help his situation or serve as an inspiration.
  Finding all concepts related to Julia sets is a reasoning task on domain knowledge, whereas finding all assertions in which these concepts occur requires reasoning on structural knowledge.

We conclude that uniting structural and domain semantics in one knowledge base and thereby making both available to reasoning systems results in superior answers to user queries compared to knowledge bases which contain domain knowledge alone.

3.5.1 Application to the Mathematical Semantic Web

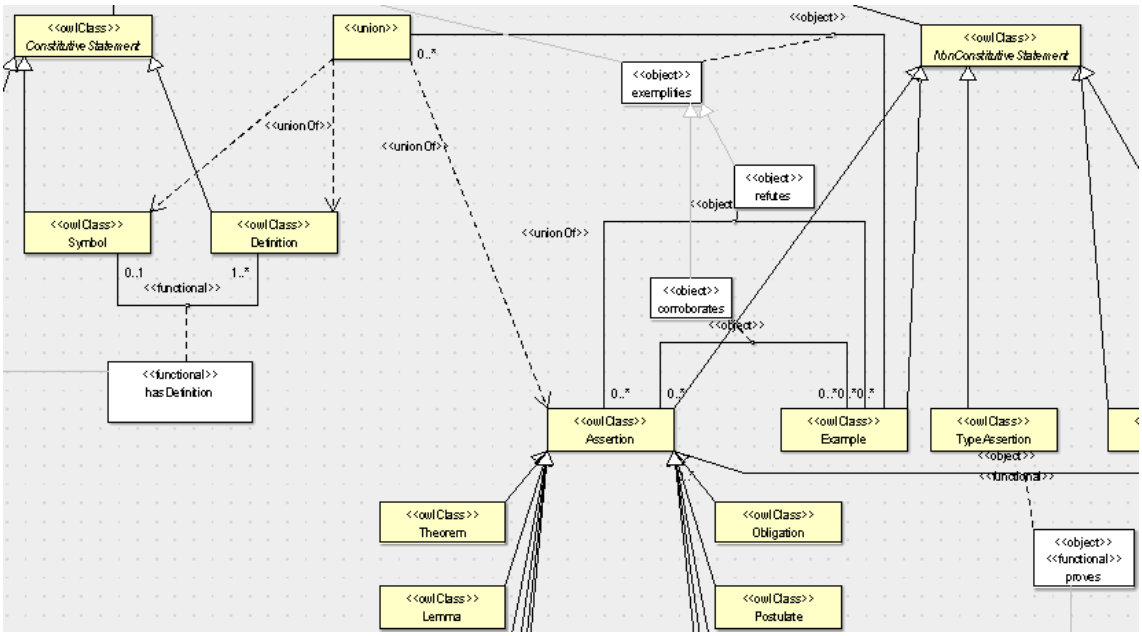

Figure 4: Extract from the OMDoc document structure ontology

The authoring process suggested in chapter 3.4 extracts the ontological statements (the formalized domain knowledge) from OMDoc, thereby effectively loosing any information about the contextwhich entails the loss of structural knowledge. In light of the analysis above we suggest a refinement of the authoring process which enriches the Mathematical Semantic Web by structural knowledge.

OMDoc provides a rich infrastructure for making the structure of a document explicit. A document structure ontology is currently being developed [Lange, 2007] which conceptualizes the document structure of OMDoc as specified in [Kohlhase, 2006]. The OMDoc document structure ontology is written in OWL DL and part of the visual representation of the ontology is shown in figure 4.

It would be desirable to relate the concepts, relations and instances captured in the Math Ontology Language and embedded in OMDoc to the structural elements in which they are defined, thereby establishing a link between domain and structural knowledge which can be exploited for reasoning.

The structural blocks in OMDoc are instances of the OMDoc document structure ontology, hence one OMDoc document is comprised of a number of instances of the various concepts defined in the ontology. These instances can be automatically extracted from an OMDoc document.
The mathematical ontology on the other hand contains concepts, relations and instances which we would like to relate to the structure ontology instances. Unfortunately, Description Logics treats concepts, relations and instances as disjoint entities which means that it is impossible to relate domain concepts to structure instances[27]. This limitation of DL disallows, for instance, to relate the concept `Group` to an instance of `Definition` (a concept of the OMDoc document structure ontology) via the relation `defined-in`.

To overcome this limitation in order to connect ontological definitions pertaining to mathematical knowledge with the structure ontology instances in which they are defined, a work around using the features of OWL DL has been developed. Instead of relations, annotation properties, which were introduced in OWL DL to annotate statements as the name suggests, are used to relate concepts to instances. Annotation properties are treated like comments and are hence not restricted by the limitations of DL but supported by Semantic Web tools.
The mathematical ontology containing annotations can be translated into another ontology in which the concepts, relations, and instances from the former one are declared as instances of the concepts *MathConcept*, *MathRelation*, and *MathInstance* respectively. The later ontology is based on the "MathRealm" ontology where the mentioned concepts are defined (see the appendix A.2 for a complete listing). Furthermore, the MathRealm ontology redeclares the annotation properties as object properties (i.e. relations) through a change in namespace. The MathRealm ontology is integrated with the OMDoc document structure ontology in that it uses the concepts defined therein as range values for the object properties. By this construction, the later ontology can be used for reasoning on domain concepts and structure instances, while the former ontology contributes to the

[27]RDFS and OWL Full allow to treat concepts as instances, but there exist no efficient reasoning algorithms for these more expressive knowledge representation languages.

mathematical knowledge base.

The translation between both ontologies establishes the connection. The translation process carries out the following steps:

1. Change the entity definition from
   `http://www.mathweb.org/2007/01/math-ontology#` to
   `http://www.mathweb.org/2007/02/math-ontology#` , thereby changing the imported ontology and the namespace definition which redeclares annotation properties to object properties (see the appendix A.2).

2. Delete all owl constructs from the ontology that do not fulfill one of the following conditions:

   - It is a declaration (construct using rdf:ID)
   - Its first child is an annotation (e.g. <mont:xyz)

3. Rename:
   - `owl:Class` to `mont:MathConcept`
   - `owl:ObjectProperty` to `mont:MathRelation`
   - everything that does not start with `mont:` to `mont:MathObject`

The entire process is visualized in figure 5. To illustrate the workings of our approach, we consider an excerpt introducing sets and proper classes from a chapter on sets and classes taken from the Graduate Algebra textbook [Hungerford, 1974] as an example. We will demonstrate all conversions on the sample excerpt. The numbers used throughout the example correspond to the numbers in figure 5.

0. The content of the chosen sample excerpt reads as follows:

Definition: A class A is defined to be a set if and only if there exists a class B such that $A \in B$. Thus a set is a particular kind of class.

Definition: A class that is not a set is called a proper class.

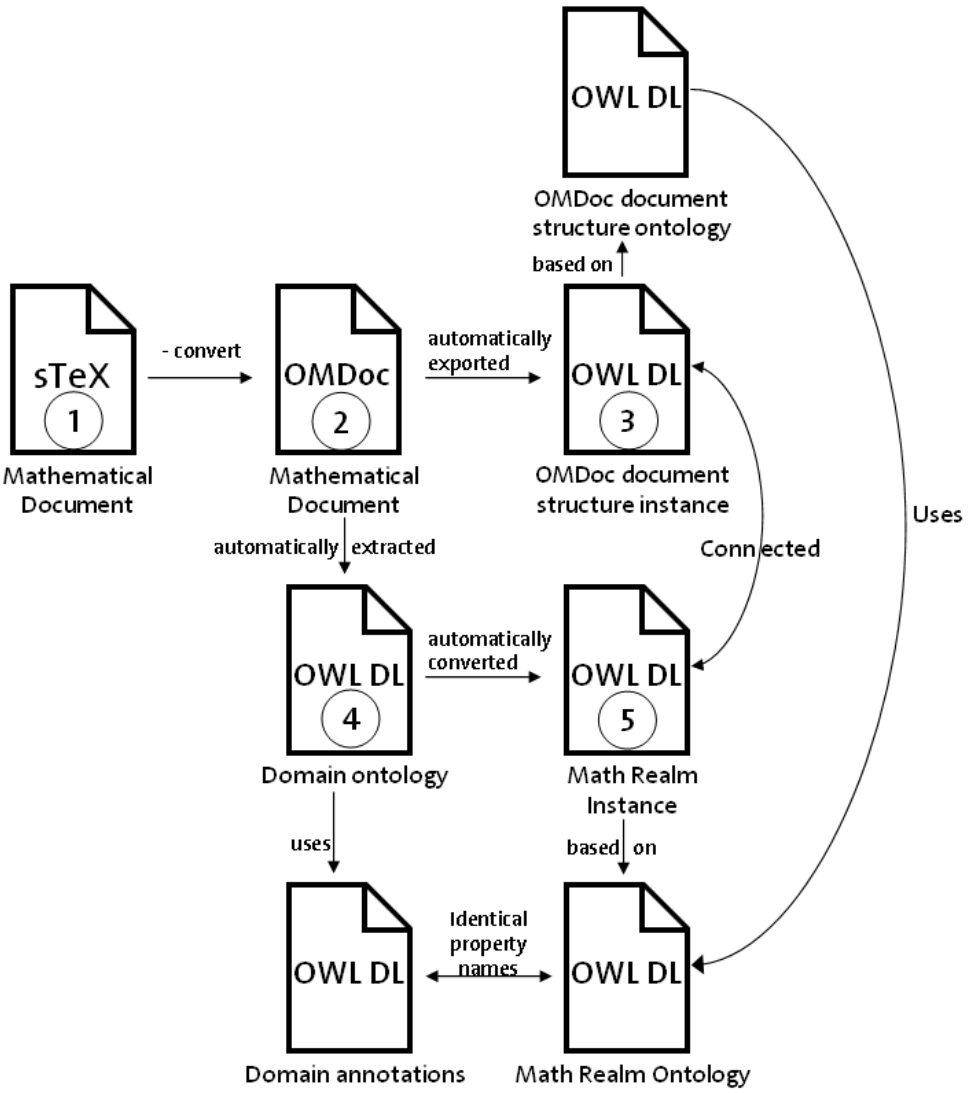

Figure 5: Ontology Extraction Process

1. The document is first written in sTEX:

```
\symdef[type=concept]{Set}{set}

\begin{definition}[id=set.def,display=block,for=Set]
  A class $A$ is defined to be a {\definiendum[Set]{set}} if and only
      if there
  exists a class $B$ such that $\inset{A}{B}$. Thus a set is a
      particular kind
  of class.
  \begin{FMP}[logic=dl]
    $\OConceptDef{\Set}{\OIntersection{\Class,\OExistsRelationWith{\
        membership}{\Class}}}$
  \end{FMP}
\end{definition}

\symdef[type=concept]{ProperClass}{proper class}

\begin{definition}[id=properClass.def,display=flow,for=ProperClass]
  A class that is not a set is called a {\definiendum[ProperClass]{
      proper
  class}}.
  \begin{FMP}[logic=dl]
    $\OConceptDef{\ProperClass}{\OIntersection{\Class,\OComplement{\
        Set}}}$
  \end{FMP}
\end{definition}
```

Listing 7: Step 1

2. Using LaTeXML the entire sTEX document is translated to OMDoc yielding the following result:

```
  <symbol name="Set">
    <type system="OntLang"><om:OMOBJ><om:OMS cd="OntLang" name="
        concept" /></om:OMOBJ></type>
  </symbol>
  <presentation for="#Set"><use format="default">set</use></
      presentation>

  <definition xml:id="set.def" for="#Set" type="simple">
    <CMP>
    A class <om:OMOBJ><om:OMV name="A" /></om:OMOBJ> is defined to be
        a
    <term cd="SetsClasses" name="Set" role="definiedum">set </term>
        if and only if
    there  exists a class <om:OMOBJ><om:OMV name="B" /></om:OMOBJ>
        such that
    <om:OMOBJ><om:OMA><om:OMS cd="SetsClasses" name="membership" /><
        om:OMV name="A" />
    <om:OMV name="B" /></om:OMA></om:OMOBJ>. Thus a set is a
        particular kind of class.
    </CMP>
```

```
    <FMP logic="dl">
      <om:OMOBJ>
        <om:OMA>
          <om:OMS cd="OntLang" name="OConceptDef" />
          <om:OMS cd="SetsClasses" name="Set" />
          <om:OMA>
            <om:OMS cd="OntLang" name="OIntersection" />
            <om:OMS cd="SetsClasses" name="Class" />
            <om:OMA>
              <om:OMS cd="OntLang" name="OExistsRelationWith" />
              <om:OMS cd="SetsClasses" name="membership" />
              <om:OMS cd="SetsClasses" name="Class" />
            </om:OMA>
          </om:OMA>
        </om:OMA>
      </om:OMOBJ>

    </FMP>
  </definition>

  <symbol name="ProperClass">
    <type system="OntLang"><om:OMOBJ><om:OMS cd="OntLang" name="
        concept" /></om:OMOBJ></type>
  </symbol>
  <presentation for="#ProperClass"><use format="default">proper class
      </use></presentation>

  <definition xml:id="properClass.def" for="#ProperClass">
    <CMP>
    A class that is not a set is called a
    <term cd="SetsClasses" name="ProperClass" role="definiedum">
        proper class </term>.
    </CMP>

    <FMP logic="dl">
      <om:OMOBJ>
        <om:OMA>
          <om:OMS cd="OntLang" name="OConceptDef" />
          <om:OMS cd="SetsClasses" name="ProperClass" />
          <om:OMA>
            <om:OMS cd="OntLang" name="OIntersection" />
            <om:OMS cd="SetsClasses" name="Class" />
            <om:OMA>
              <om:OMS cd="OntLang" name="OComplement" />
              <om:OMS cd="SetsClasses" name="Set" />
            </om:OMA>
          </om:OMA>
        </om:OMA>
      </om:OMOBJ>
    </FMP>
```

```
</definition>
```

Listing 8: Step 2

3. The structure elements of the OMDoc document (in this example `Symbol` and `Definition`) are instances of the OMDoc document structure ontology. These instances are automatically extracted from the original OMDoc document using XSLT stylesheets and declared in OWL DL. The sample excerpt yields the following document structure ontology instances:

```
<omdocOnt:Symbol rdf:ID="Set">
  <omdocOnt:hasDefinition rdf:resource="set.def" />
</omdocOnt:Symbol>

<omdocOnt:Definition rdf:ID="set.def">
</omdocOnt:Definition>

<omdocOnt:Symbol rdf:ID="ProperClass">
  <omdocOnt:hasDefinition rdf:resource="properClass.def" />
</omdocOnt:Symbol>

<omdocOnt:Definition rdf:ID="properClass.def">
</omdocOnt:Definition>
```

Listing 9: Step 3

4. In a similar manner, the ontological definitions written in the Math Ontology Language are extracted into an OWL DL file including annotations which relate the concepts to the document structure instances.:

```
<owl:Class rdf:ID="Set">
    <mont:declaredIn rdf:resource="&orgdoc;SetsClasses" />
</owl:Class>

<owl:Class rdf:about="#Set">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Class" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="#membership" />
          <owl:someValuesFrom rdf:resource="#Class" />
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#Set">
  <mont:definedIn rdf:resource="&orgdoc;set.def" />
</owl:Class>

```

```
<owl:Class rdf:ID="ProperClass">
    <mont:declaredIn rdf:resource="&orgdoc;SetsClasses" />
</owl:Class>

<owl:Class rdf:about="#ProperClass">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Class" />
        <owl:Class>
          <owl:complementOf>
            <owl:Class rdf:about="#Set" />
          </owl:complementOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#ProperClass">
  <mont:definedIn rdf:resource="&orgdoc;properClass.def" />
</owl:Class>

<owl:Class rdf:about="#Set">
  <mont:occursIn rdf:resource="&orgdoc;properClass.def" />
</owl:Class>
```

Listing 10: Step 4

5. Finally, the above OWL DL ontology is translated to an ontology based on "MathRealm" which is used to relate the knowledge objects to the structure elements in which they occur.

```
<mont:MathConcept rdf:ID="Set">
    <mont:declaredIn rdf:resource="&orgdoc;SetsClasses" />
</mont:MathConcept>

<mont:MathConcept rdf:about="#Set">
  <mont:definedIn rdf:resource="&orgdoc;set.def" />
</mont:MathConcept>

<mont:MathConcept rdf:ID="ProperClass">
    <mont:declaredIn rdf:resource="&orgdoc;SetsClasses" />
</mont:MathConcept>

<mont:MathConcept rdf:about="#ProperClass">
  <mont:definedIn rdf:resource="&orgdoc;properClass.def" />
</mont:MathConcept>

<mont:MathConcept rdf:about="#Set">
  <mont:occursIn rdf:resource="&orgdoc;properClass.def" />
</mont:MathConcept>
```

Listing 11: Step 5

To close this section we present the refined and complete authoring process in figure 6.

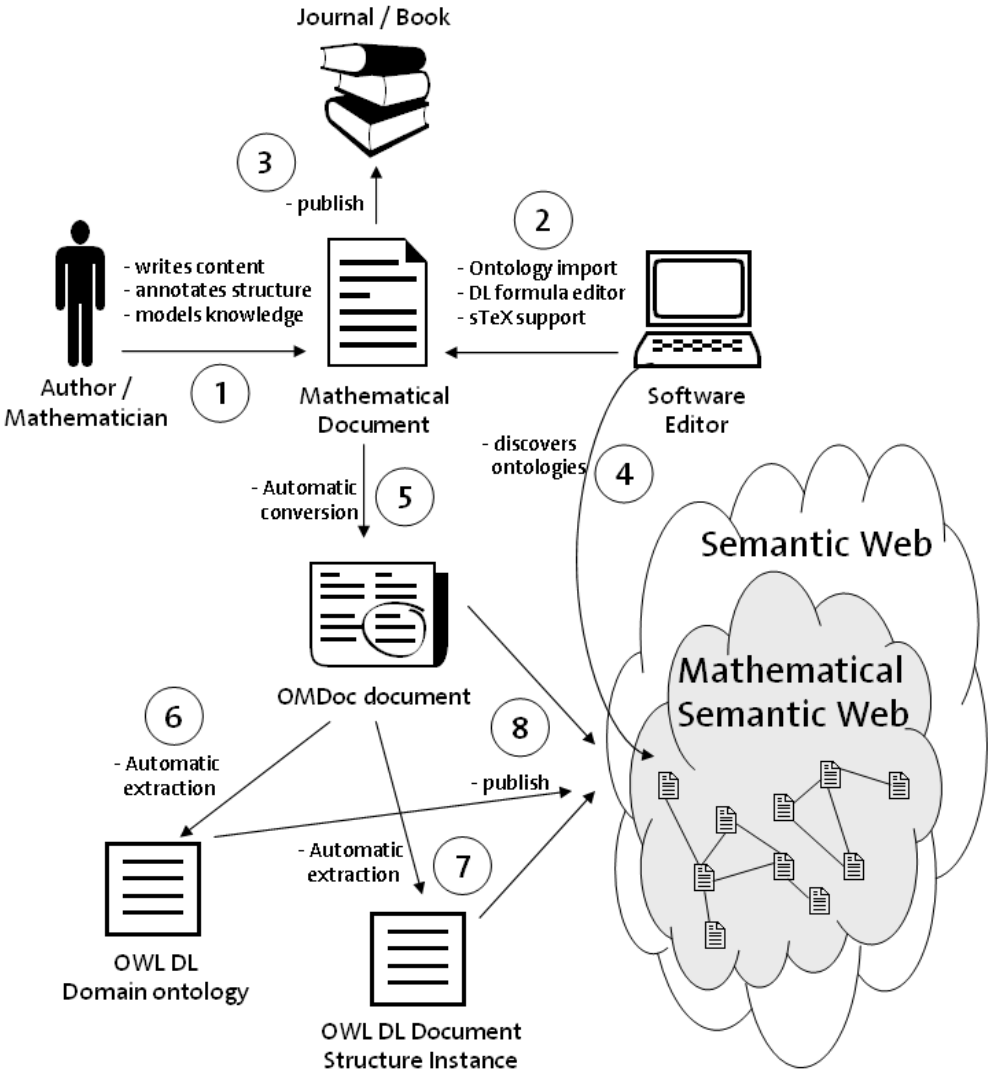

Figure 6: Final Authoring Process on the Mathematical Semantic Web

3.6 Mathematical Semantic Web Services

This chapter has presented the technical underpinning of the Mathematical Semantic Web in detail, but not yet provided incentives for mathematical authors to contribute to the gloabl mathematical knowledge base we propose by capturing the knowledge of their documents in the Math Ontology Language.
The ontology annotation requires an initial investment by the document author which must ultimately be rewarded by useful services which feed on the knowledge that is captured in the resulting ontologies. In the following we describe some possible services for the Mathematical Semantic Web, but this list is by no means meant to be exhaustive.

3.6.1 Advanced Search

An advanced search engine operates on the OWL DL ontologies which aggregate to the knowledge base of the Mathematical Semantic Web and searches ABox as well as TBox knowledge. It is important to understand the difference between semantic search and keyword-based search which dominates today's web. Semantic search can draw inferences on explicitly represented knowledge and thereby approximates actual knowledge search.

An ABox search retrieves mathematical objects (instances) on user request, such as *"A ring without a* 1*"*, *"A continuous bijection which is not a homeomorphism"*, *"A projective module over some ring* R*"*. The search engine searches for instances of the respective concepts and report those objects which have either been directly defined as concept instances or for which concept membership can be inferred.
For example, an author might have declared a certain ring, which he introduced in his document, to be without a 1. On the other hand, membership can also be inferred. For instance, one author might have introduced a free module F over a specific ring R and declared it as such in his ontology. Another author proved that every free module over a ring with identity is projective and as a consequence equated the concepts in the ontology. The search engine infers that F is projective (given that R has an identity) and return it to the user.
Ignoring efficiency considerations, such a search engine could be build by providing an intuitive user interface to the *retrieval* service of Description Logic reasoners (see chapter 1.3.4).

The TBox search on the other hand retrieves defined mathematical concepts which match the user query and displays them together with the relationships to other concepts. For example, the query *"holomorphic function"* would return the concept of a holomorphic function and an extract of the function taxonomy showing, for instance that the concept is identical to that of an analytic function. TBox search could be implemented using the *classification* service of Description

Logic reasoners (see chapter 1.3.4).

In section 3.5 we propose to unite domain and structural semantics to allow for reasoning over both types of knowledge in one user query. Search engines build on this theory [28] can more accurately determine what the user is looking for as was shown in the examples from section 3.5 which are repeated here for convenience:

- **Finding examples**
  A mathematics teacher would like to post a homework questions which asks the students to prove that a certain mathematical object is a group. Hence the teacher is interested in retrieving all documents which introduce instances of the concept `group` in an example.
  Finding all instances of the concept `group` requires reasoning on domain knowledge, whereas retrieving documents in which these instances occur as an example is accomplished through reasoning on structural knowledge.

- **Finding theorems**
  A mathematician is conducting research on Julia sets and currently does not progress on a particular proof. Hence he would like to retrieve all assertions made about Julia sets and related concepts to find out whether any of these could help his situation or serve as an inspiration.
  Finding all concepts related to Julia sets is a reasoning task on domain knowledge, whereas finding all assertions in which these concepts occur requires reasoning on structural knowledge.

- **Finding definitions**
  A mathematics student is reading a paper which presumes knowledge about a concept that the student has never heard of. Hence he would like to find documents which define and describe the concept in question and all related concepts.
  Finding all related concepts[29] to the concept in question is achieved through reasoning on domain knowledge, while retrieving documents that define and/or explain these concepts requires reasoning on structural knowledge.

Due to its compatibility with the Semantic Web, the Mathematical Semantic Web can reuse semantic search engines which are currently being developed and therefore benefit from research and development in this area. Swoogle[30]

[28] Reasoning over domain and structural knowledge requires a two step reasoning process - one for each knowledge base. The reasoning steps are connected via the translation described in section 3.5.1. The two step reasoning process relies on the availability of nominals which are indeed included in OWL DL. Unfortunately though, popular reasoning systems such as RacerPro^{TM} do not yet support nominals, but given that OWL DL is the official Description Logic language for the Semantic Web this should change in the near future.

[29] "All related concepts" is understood as the union of all concepts to which the concept in question is explicitly related via defined relations and those concepts which are in close proximity in the taxonomy tree

[30] http://swoogle.umbc.edu/

[Ding et al., 2005] is one such search engine, whose index could be extended to cover the Mathematical Semantic Web[31].

3.6.2 Visual Conceptualization

To get an initial understanding or a comprehensive overview of a particular branch of mathematics it is helpful for the human user to see all concepts defined therein displayed in a visual presentation. Hence a visual conceptualization of mathematics improves learning and fosters a deeper understanding of the material. This fact has been recognized in mathematical education in that some authors provide taxonomy diagrams with their expositions (e.g. [Hungerford, 1974]).

On the Mathematical Semantic Web visual conceptualizations can be generated automatically by relating the concepts of interest through the ontologies in which they are defined (this is achieved through the *classification* reasoning service of DL systems). Prototypes for ontology visualization already exist and are highly customizable. The Protégé framework [Knublauch et al., 2004], for example, features various plugins for visualization. Extensive literature [Akrivi et al., 2006], [Storey et al., 2004] on this topic provides a starting point for the evaluation of visualization techniques for the Mathematical Semantic Web.

3.6.3 Mathematical Tutoring Systems

Mathematical tutoring systems can be based on the comprehensive ontologies of the Mathematical Semantic Web. Tutoring systems[32] retrieve information using the semantic search service outlined above and present it to the user. They track the users understanding of mathematical concepts which allows for efficient interaction with mathematical documents. When the user opens a document, the tutoring system can automatically determine which concepts are being introduced in the document and which are presupposed based on the ontological statements contained therein. This information is then matched with the user's record. If the tutoring system concludes (based on the matching results) that the user is unlikely to understand the current document because she is lacking knowledge in this field, the system alerts the user and automatically suggests documents to be read prior to the one at hand.

3.6.4 Classification of mathematical documents

Yet another service utilizing the created mathematics ontology could be that of mathematical document classification. This service allows a user to browse

[31] A search for "holomorphic" currently returns 0 results

[32] Web-based, user-adaptive ActiveMath learning environment for mathematics (homepage at http://www.activemath.org/) already implements many of the mentioned features and could be adapted for the Mathematical Semantic Web.

through the areas of mathematics at different levels of granularity and display all articles, papers, and books that address the specific area of interest. For example, if one's interest lies in oriented manifolds, the service could be used to retrieve a list of mathematical documents that deal with these objects.
Mathematical documents are automatically associated to the respective areas based on the corresponding, extracted ontological definitions. Classification based on ontologies is much more accurate than simple keyword based retrieval, because it only associates documents to the concepts actually addressed.
This service would feed on both the domain and structural knowledge to classify the document and hence follow the two step reasoning process.

As an example, let us consider the Mathematics Subject Classification 2000 (MSC2000) [Society, 2000] maintained by the American Mathematical Society. The Mathematics Subject Classification is used to categorize items covered by the two reviewing databases, Mathematical Reviews (MR) and Zentralblatt MATH (Zbl)[33]. MSC2000 systematically assigns to each area of mathematics a 5 digit identifier. A human reviewer then determines a "primary" classification based on what she deems to be the most appropriate subject area.
On the Mathematical Semantic Web document classification can be generated automatically. Concepts can be assigned to categories using annotation properties and hence the document can be classified based on the concepts it contains. This requires a one time investment on side of the classifying body (much like the development of MCS2000 did) and can be easily extended as mathematical knowledge growths.

[33]compare http://www.ams.org/msc/

4 Mathematical Ontology Engineering

The previous chapter introduced the general architecture, the individual building blocks and possible tool support of the proposed Mathematical Semantic Web. Using the technical foundation, this chapter addresses the creation of mathematical ontologies from the perspective of an author. Best practices for mathematical ontology engineering are derived from a thorough literature review and the author's own experience.

4.1 Existing Mathematical Ontologies

Prior to engaging in an independent analysis on mathematical ontology engineering we review previous work on mathematical ontologies in order to benefit from lessons already learned. To our knowledge, three projects have conducted research in this area which are described in the following and compared to our work by outlining the differences and similarities[34].

4.1.1 EngMath

EngMath [Gruber and Olsen, 1994] is an ontology for engineering mathematics which "includes conceptual foundations for scalar, vector, and tensor quantities, physical dimensions, units of measure, functions of quantities, and dimensionless quantities". Despite its constrained application domain the authors anticipated an extension of their ontology to other theories of mathematics and regard their work as a conceptual foundation. On the other hand, most of the results which pertain to the design techniques of mathematical ontologies are very specific to mathematical engineering.
The primary purpose of the EngMath ontology is the facilitation of agent communication in that cooperating agents commit to the EngMath ontology and can therefore exchange mathematical knowledge in the form of queries, assertions or general statements. Consequently, the EngMath ontology can be considered as a shared vocabulary, since it defines the meaning of terms to which agents commit. The actual meaning of the terms defined remains unaccessible to agents, since the semantics is given in Lisp notation which cannot be used for efficient reasoning. The following listing displays the definition of `Abelian Group` in the EngMath Ontology [35].

[34] One might argue that the MMiSS project [Mahnke and Scheffczyk, 2005] is missing from the list, since it uses ontologies of mathematical terms for consistency management of documents. But these ontologies are used solely for the purpose of defining a contract that a document commits to vis-a-vis other documents and do not aim at capturing the actual knowledge of the application domain.

[35] retrieved from: http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/abstract-algebra/ABELIAN-GROUP.html on May 2nd, 2007

```
Relation ABELIAN-GROUP
    * Defined in theory: Abstract-algebra
    * Source code: abstract-algebra.lisp

Slots on this relation:
  Arity: 3
  Subrelation-Of: Group

Axioms:
    (<=> (Abelian-Group ?Domain ?Op ?Id)
         (And (Group ?Domain ?Op ?Id) (Commutative ?Op ?Domain)))
```

Listing 12: Abelian Group definition from EngMath Ontology

The example demonstrates that the knowledge available for reasoning is restricted to the fact that abelian groups are a sub concept of groups[36]. Knowledge about commutativity is hidden in the axiom. The axiom is formalized but in a language that is more expressive than First Order Logic, hence it is appropriate to speak of hidden knowledge as this fact is unaccessible to efficient reasoning systems. The ontology of mathematics we propose tries to fully define concepts within the ontology and thereby making all knowledge about concepts explicit and accessible to reasoners. Compare a possible definition of `Abelian Group` in the Math Ontology Language as given in 6 to the listing above. Besides the different

AbelianGroup=	Group $\sqcap$ $\exists$ operation.(AssociativeOp $\sqcap$ InverseOp $\sqcap$ IdentityOp $\sqcap$ CommutativeOp)

Table 6: Definition of abelian group

modeling approach (intensional versus extensional), the definition in the Math Ontology Language has much richer machine readable semantics. On the point of similarity, EngMath and the proposed Math Ontology Language share the design goal of being applicable to all areas of mathematics.

4.1.2 MONET

The principal objective of the MONET project [Caprotti et al., 2004] is to develop a framework for the description and provision of web-based mathematical services. The MONET ontologies are designed to model queries and service descriptions in order to use reasoning systems to match user queries to offered mathematical services. Much like the EngMath ontology, the MONET ontologies establish a common communication language for agents on the web and only declare mathematical objects without explicating their meaning. Hence we arrive at the same conclusion, namely that the MONET project does not offer any valuable insights

[36]The reader shall not be confused by the fact that the EngMath ontology uses the word "relation" for what we have called "concept" before.

on creating an ontology of mathematics which fully captures the meaning of the concepts defined therein. On a technical level though, the MONET project builds upon the same technology as the Mathematical Semantic Web and therefore we can benefit from the experiences made and are aware of the shortcomings observed (e.g. see [Caprotti et al., 2004, Chapter 2].
Furthermore, one should note that both, the EngMath and the MONET ontologies, are designed and maintained by one authority in order to fulfill their primary purpose of constituting a shared vocabulary for (web) agents. This centralized approach is in sharp contrast to the decentralized Mathematical Semantic Web we envision. For this reason, the ontologies created on the Mathematical Semantic Web cannot serve as a shared vocabulary directly although they can be used to facilitate agent communication (see, for example, 4.4.1) and certain parts could be standardized to accommodate this purpose[37].

4.1.3 Projective Geometry Ontology

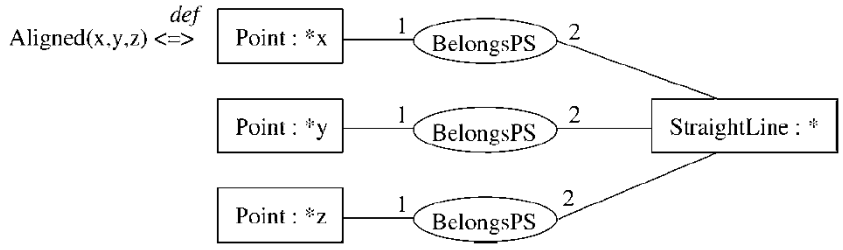

Figure 7: Example of a Conceptual Graph

In their work [Fürst et al., 2003] Fürst et al. describe the formalization of the ontology underlying projective geometry and discuss ontology engineering in the context of mathematical knowledge management. In principle their approach is similar to ours in that they also aim at capturing the semantics of the mathematical objects defined in the ontology to allow for reasoning in the theory of projective geometry. In contrast, their ontology contains enough knowledge to serve as a knowledge base for theorem provers (compare to chapter 2.4), which is due to the expressiveness of the logical formalism employed, namely Conceptual Graphs. Conceptual Graphs can be considered as a graphic notation of the logical theory expressed in First Order Logic and as such enjoys the same expressiveness [Fürst et al., 2003, Chapter 2.2]. The authors motivation for using Conceptual Graphs as the formalism for their knowledge base is that they are allegedly easier to read. While this argument seems intuitive, as it is common to state easier comprehension as the primary advantage of visual representations, it is hard to follow for logical formalisms, especially beacuase the authors declare the users to be experts. For example, it is not obvious why the Conceptual Graph in figure

[37] A standardization would of course require the involvement of a central authority.

7 is easier to read than the equivalent formula below, in particular, because the graph contains some implicit knowledge[38].

$\forall$ Point(p1),Point(p2),Point(p3): Aligned(p1,p2,p3) $\Leftrightarrow$
($\exists$ StraightLine(l): BelongsPS(p1,l) $\wedge$ BelongsPS(p1,l) $\wedge$ BelongsPS(p1,l))

We argued before (see 2.3.2) that the complexity of a logical formalism comes as a tradeoff with its expressiveness which also applies to Conceptual Graphs (CGs). In fact, modeling with CGs does involve bound variables and variable scoping (compare 2.3.2). Hence, the work on formalizing the ontology underlying projective geometry is comparable to other formalization efforts as described in chapter 2.2 and therefore suffers from the same shortcomings and advantages compared to the Mathematical Semantic Web we propose.

[38]The variable scoping in Conceptual Graphs is not directly clear and might cause confusion depending on the layout of the graph.

4.2 Ontology Engineering Methodologies

While the previous section reviews existing work on ontologies of mathematics, this section provides an overview of general ontology engineering methodologies which apply to any application domain. The knowledge management community has produced a vast body of literature which discusses various approaches to developing an ontology. We introduce some popular methodologies and analyse their applicability to the Mathematical Semantic Web. The best practices distilled in this process are summarized in section 4.3.

[Nagypál, 2005] describes a reference ontology building methodology based on the analysis of existing state-of-the-art methodologies by combining elements from METHONTOLOGY and the On-to-Knowledge methodology, which are the two most mature methodologies according to study [Fernández-López and Gómez-Pérez, 2002]. The second methodology considered, namely [Noy and McGuinness, 2001], is less elaborate in that it does not detail knowledge acquisition processes but offers a 7 step plan and some suggestions for common design decisions. Despite the difference in the level of detail, both methodologies are in general very similar and therefore treated simultaneously in the following.

Ontology building methodologies have reached the maturity and also the complexity of software engineering methodologies and hence the analogy between ontology engineering and software engineering is commonly drawn in both expositions.
To assess the applicability we follow the phases of the overarching process common to both methodologies. The multitude of other processes described are either briefly analysed thereafter or simply ignored, because they do not pertain to ontology development on the Mathematical Semantic Web; like, for example, project management or life cycle activities.

4.2.1 Specification

During the initial phase, the domain, scope and goal of the ontology are defined. Competency questions help to close in on the scope of the ontology. Furthermore reuse of existing ontologies is evaluated and information sources listed.

In a way, this entire document can be considered as a specification of the global ontology of mathematics build on the Mathematical Semantic Web. The domain of the ontology is mathematics and the scope is left unconstrained in that the ontology aims at eventually covering all theories of mathematics. Possible uses and consequently goals have been specified in previous chapters. For example, the service descriptions in chapter 3.6 can be interpreted as competency questions, because they implicitly define the kind of questions to which the ontology should provide answers.

On the other hand, providing an exact specification as the methodologies suggest could be considered counter productive, because it would essentially limit the uses to which the ontology can be put. In other words, the ontology of mathematics we propose is dynamic and ever evolving, hence a specification would be outdated by design, because sources of information or authors are changing frequently.
Reusing existing ontologies is evaluated in section 4.1.

4.2.2 Conceptualization

The conceptualization phase is subdivided into steps which are performed in an iterative manner to arrive at the conceptual model of the application domain. Different strategies can be employed in this phase, but it is common to start with an enumeration of the terms, verbs and constants in the ontology as a first step. Based on the initial vocabulary, concepts are defined and structured in a concept hierarchy before relations between the concepts are added in step three. Finally, concept instances are discovered.
Conceptualization is analogous to analysis and design in software engineering [Sommerville, 2004] and considered to be the most important phase by most ontology engineering methodologies, since they propose a multitude of supporting steps, guidelines and evaluation criteria. Consequently, this phase unfolds into a complex and time intensive process.

Conceptualization addresses the difficulty of elicitating knowledge from natural language sources or human understanding which often times requires making implicit knowledge explicit.

Treatments of mathematics, on the other hand, are semi formal documents, since the mathematical vernacular (compare 3.2) used therein exhibits some characteristics of logical formalisms. In particular, it is common practice to introduce mathematical objects via definitions and to state facts about these objects in the form of theorems and lemmas.
In line with this observation, we argue that mathematical documents are inherently conceptualized and conclude that the conceptualization phase is superfluous for mathematical ontology engineering.
The author of mathematics is aware of the concepts discussed in her work, since they are either imported from other theories or explicitly defined. The definitions may contain relations to other concepts and provide all the information needed to derive a full conceptual model. Theorems and lemmas contribute further knowledge to the ontology and instances are usually given as examples or exercises. Therefore one can consider conceptualization to be a common mathematical practice, a point which is investigated in section 4.3.4.

We are aware that it is commonly argued (see, for instance, [Fürst et al., 2001]) that knowledge elicitation is an integral part of formalizing mathematical theo-

ries, since a significant share of the knowledge is left implicit in mathematical documents. While this is a valid point for the construction of knowledge bases to be used with theorem provers, it does not apply to the ontology of mathematics we propose, since the primary users are humans capable of supplementing implicit knowledge. Best Practice 4.3.5 explains in more detail why the ontology can afford that some knowledge is left implicit.

4.2.3 Formalization

The conceptual model is finally formalized in the knowledge representation language of choice during the formalization phase. This phase is analogous to implementation in software engineering and an actual formalization process depends on the language chosen[39].

Given that mathematical discourse is already conceptualized, mathematical ontology engineering basically reduces to a translation from mathematical vernacular into the Math Ontology Language described in chapter 3.2.1. Section 4.3 recommends best practices which simplify this formalization effort.

4.2.4 Maintenance

As mentioned in chapter 3.2 ontologies are usually developed independently from the knowledge sources (e.g. documents) on which their definitions rely. This entails that any modification of the original knowledge source could require a corresponding update of the ontology. Consequently, management of change in particular and maintenance in general are important to ensure soundness of ontologies after their development.

The ontology format suggested in 3.2 allows for ontological statements inside mathematical documents, in other words, inside the knowledge source. This fact makes administering change management simple, since affected ontology elements can be determined upon modification using the document structure ontology (see 3.5).

4.2.5 Design Principles

In addition to ontology development processes both methodologies describe general ontology design principles and important observations which are partly summarized and commented on below:

[39] [Nagypál, 2005] differentiates between formalization and implementation which stems from an alleged difference between general and specific formal languages. Since automatic translations between the two exist this difference is considered negligible and the two phases are merged into one.

- **There is no one correct way**:
  It is a general observation from knowledge representation that there are multiple correct ways of defining concepts in an ontology - a circumstance which is addressed in best practice 4.3.1.

- **Ontology engineering is an iterative process**:
  Ontology engineering is described as an iterative process analogous to software engineering in the sense that each iteration yields a more accurate approximation of the desired result. Unless the mathematical theory is still under development, such iterations are not necessary for ontology engineering for mathematics, since the ontological statements are a direct translation of the respective definitions, theorems and lemmas. Best practice 4.3.8 advocates an alternative to iterations, namely ontology refinements.

- **Optimal number of subconcepts**:
  Both methodologies suggest that the optimal number of subconcepts ranges from 2 to 12. While it is obvious that such an arbitrary limitation is not necessary for an ontology of mathematics, this design principle is representative for many similar ones which help the ontology engineer to develop an intuition for the conceptualization process. We argue that conceptualization is inherent to mathematical practice and that building an intuition about mathematical objects, their characteristics and relations is implicitly achieved through education in mathematics.

- **Documentation**:
  Ontology documentation is important for maintenance activities and ontology reuse because it helps the reader understand the ontological statement and points to other sources of information from which statements derived.
  The ontology format we propose (see chapter 3.2) reduces the documentation effort to a minimum, because the ontological statements are directly appended to the text fragments in the original document verbalizing the logical formula in mathematical vernacular. Ontology documentation can therefore be limited to explanations of the rationale behind particular formalizations of the author.

- **Reuse**:
  The methodology described in [Nagypál, 2005] emphasizes ontology reuse which is also of great importance to the success of the Mathematical Semantic Web as argued in chapter 3.3.2. Due to its significance for our work we dedicate section 4.4 to this topic.

- **Modularization**:
  The modularization of large ontologies facilitates reuse and simplifies maintenance. We argue that the global ontology of mathematics is divided into

modules based on the structure of the documents in which they are contained. The structure of a document makes the individual blocks explicit and allows to split ontologies naturally into modules. With the `theory` element the OMDoc format provides functionality for this specific purpose. Theories in OMDoc are build on (i.e. import) other theories and are otherwise assumed to be self contained. Thus, theories lend themselves to define ontology modules.

4.2.6 Conclusion

Despite some design principles which have influenced the formulation of best practices in section 4.3, we conclude that existing ontology engineering methodologies offer little support for the development of a global ontology of mathematics for mainly two reasons.
Firstly, the ontology engineering methodologies reviewed above target application domains which are significantly less formalized than mathematics and therefore more ambiguous when it comes to knowledge acquisition. This fact alone motivates many of the processes and principles proposed which are consequently irrelevant for mathematical ontology engineering.
Literature on ontology engineering commonly differentiates between an ontology engineer and a domain expert whereas the Mathematical Semantic Web rests on the assumption that a mathematical author accepts both roles due to the ease of knowledge modeling in the Math Ontology Language and extensive tool support.

More fundamentally though, existing ontology engineering methodologies are heavily influenced by their counterparts from software engineering in that can be considered as translations of established practices from software development to ontology development. Consequently, they inherit a comparable complexity and are based on the same assumptions. Ontology engineering methodologies presuppose the existence of a well defined group of ontology engineers and a central authority which is required to coordinate the development activities and supervise the processes.

The Mathematical Semantic Web proposed does not satisfy these assumptions, because it has been designed specifically without central authority welcoming contribution from any author. Recent proposals of agile ontology engineering methodologies (e.g. RapidOWL [Auer, 2006]) try to overcome the stringent processes described in earlier frameworks and accommodate for the highly volatile status of a heterogeneous, ever evolving ontology, but are ultimately based on the same assumption, namely the existence of a coordinating body. To our knowledge, managing a dynamic, non-standardized, heterogeneous and decentralized ontology is an unsolved problem of the Semantic Web and subject to ongoing research.

The Wiki approach to knowledge management, implemented in the popular

online encyclopedia Wikipedia[40], is currently the most appropriate paradigm for our purposes. It neither defines processes nor imposes strict rules on the authors and hence allows everybody to freely contribute. On the other hand, it does suggest best practices for the creation of articles which are generally followed.

[40]http://www.wikipedia.org/

4.3 Best Practices

As stated in the previous section, ontology building methodologies describe complex processes and detail many individual steps. Obviously, the development of ontologies on the Mathematical Semantic Web with its diverse group of authors cannot follow any elaborate methodology due to the lack of authority and coordination. Nevertheless, valuable suggestions can simplify the ontology development to a large extent and are therefore captured in the form of Best Practices which we understand as recommendations for the mathematical author wishing to contribute.

The following lists Best Practices derived from existing methodologies introduced in section 4.2 and based on the experience gained during the annotation of the Bourbaki Algebra 1 textbook [Bourbaki, 1998] and other mathematical documents.

4.3.1 There is more than one correct Definition

The first best practice is more of a general observation than a suggestion for knowledge engineering. It clarifies that in many scenarios more than one correct formalization of a mathematical concept exists. First of all, this seeming ambiguity stems from the fact, that different definitions for the same concept are provided in the theory, which are often times proven to be equivalent. For example, at least four different definitions for a differentiable submanifold of $\mathbb{R}^N$ exist.
Hence, it is up to the author to decide which definition she uses in her work and therefore which definition to formalize in the Math Ontology Language. This choice should be made in anticipation of the context in which the concept is used.

Secondly, alternative formalizations may arise due to the limited expressiveness of the Math Ontology Language as discussed in chapter 2.4. Some properties of mathematical objects, like associativity or commutativity, cannot be formalized and hence must be introduced as atomic concepts or through relations. In chapter 2.4 two alternative definitions of the concept of differentiable functions on $\mathbb{R}$ are discussed in more detail.
As before, the author is to decide which formalization most accurately reflects the intended meaning and therefore integrates nicely with ontological statements in which it occurs. Two alternative definitions of the concept `Monoid` are given in table 7 and 8. The first definition regards "having an identity element" to be an intrinsic feature of the operation of a monoid, whereas the second definition relates identity elements to monoids directly. While the first definition is closer to those found in standard textbooks, judging on the appropriateness is impossible without knowing the specific context.

$$\text{Monoid} := \exists \text{base-set.Set} \sqcap \exists \text{ operation.}(\text{AssociativeOp} \sqcap \text{IdentityOp})$$

Table 7: Monoid Definition 1

$$\text{Monoid} := \exists \text{base-set.Set} \sqcap \exists \text{ operation.}(\text{AssociativeOp}) \sqcap \exists \text{identity.IdentityElement}$$

Table 8: Monoid Definition 2

4.3.2 Naming Conventions

Naming conventions improve the readability and ease of developing an ontology. Readers can directly infer the type of an object and other useful information which aids comprehension, whereas authors can rely on an agreed standard for choosing names. The following list summarizes the naming convention we suggest for the Mathematical Semantic Web.

- **Telling names**
  In almost all cases the mathematical concepts introduced in the ontology have been given a name in the respective theory or document, such as *group*, *manifold*, or *integrable function*. In the great minority of cases, where a concept is only implicitly contained in the document but made explicit in the ontology, the author shall use a telling name, i.e. names which suggest the intended meaning. In particular, the use of abbreviations is discouraged.

- **Capitalization**
  - Concept names shall be written in upper CamelCase[41].
  - Relation names shall be written in lower case with dash delimiter[42].
  - Instance names shall be written in lower CamelCase[43].

- **Concept names in singular**
  We made the somewhat arbitrary decision to use the singular form of concept names, so `Group` instead of `Groups`.

- **Relation names**
  Names of relations shall express how the domain and range relate to each other. Since relations are less frequently stated in mathematical documents than concepts the author has more freedom in choosing appropriate names. In many ontologies, relation names carry the prefix "has" or the suffix "of", as, for instance, in *has-derivative* or *subset-of*. The author is encouraged

[41]To arrive at the upper CamelCase version of a name, one concatenates the individual capitalized words. For example *integrable function* is transformed into *IntegrableFunction*.

[42]Using the dash character ("-") as a delimiter effectively means replacing all spaces between words by a dash. For example, *identity element* becomes *identity-element*.

[43]Lower CamelCase differs from upper CamelCase only in the first character which is lower case in the former. For example, *Koch snowflake* is written as *kochSnowflake*.

to use pre- or suffix names only if the direction of the relation can not be inferred from the name only. For example, in definition 7 it is not necessary to call the relation *has-operation*.

4.3.3 No Unique Name Assumption

It is unavoidable that the same mathematical concept is defined many times on the Mathematical Semantic Web under different names. Hence it is absolutely vital that no unique name assumption is made. The unique name assumption says, that two different names necessarily denote different concepts, relations, or instances.
In some cases this circumstance might yield unexpected results, since reasoning systems may conclude, for example, that two instances defined in the same document are actually identical rather than issuing an error for the violation of a functional constraint on a relation. The author is therefore encouraged to explicitly declare concepts to be disjoint and instances to be different from one another (using the Math Ontology Language constructs `ODisjoint` and `ODifferentFrom` respectively).

4.3.4 Capturing Mathematics Extensionally

In section 4.2.2 we argue that mathematical discourse is inherently conceptualized. In order to make the conceptualization explicit in the document, we propose to write mathematical definitions extensionally, which means to specify all objects which constitute the definition [44]. In practice, this amounts to structuring documents into theories and to state the theory constitutive elements explicitly. A theory is usually defined by a set of *symbols* (the signature) that denote the mathematical objects under consideration and a set of *axioms* describing their relationships. We refer to symbols and axioms jointly as theory constitutive elements[45].
This approach has the advantage, that symbols can be used simultaneously in the document and the ontology thereby tightly integrating the two. Furthermore, is simplifies formalization as the ontological statements are in direct correspondence to statements made in the context of a theory which reduces the translation effort.

The example given below introduces the theory of semigroups to illustrates this approach. Note in particular, that all theory constitutive elements are explicitly stated in the theory, combined in the definition and also used in the ontology statement which integrates smoothly into the theory.

```
<theory xml:id="semigroup">
```

[44]The opposite of extensional definitions are intensional ones, which refer to more general definitions and add further characteristics. For example, an intensional definition for the concept *monoid* could be: Monoid(M,∘) :⇔ (Semigroup(M,∘) ∧ Identity(M,∘)
[45]Compare to [Kohlhase and Rabe,]

```
<symbol name="Semigroup">
  <type system="OntLang"><om:OMOBJ><om:OMS cd="OntLang" name="
      concept" />
  </om:OMOBJ></type>
</symbol>


<symbol name="base-set">
  <type system="OntLang"><om:OMOBJ><om:OMS cd="OntLang" name="
      relation" />
  </om:OMOBJ></type>
</symbol>
<presentation for="#base-set"><use format="default">S</use></
    presentation>

<symbol name="AssociativeOperation">
  <type system="OntLang"><om:OMOBJ><om:OMS cd="OntLang" name="
      relation" />
  </om:OMOBJ></type>
</symbol>
<presentation for="#AssociativeOperation"><use format="default">\
    circ</use>
</presentation>

<axiom xml:id="closed.ax"><FMP>\forall x, y \in M.x \circ y \in M</
    FMP></axiom>
<axiom xml:id="assoc.ax">
  <FMP>\forall x, y, z \in M.(x \circ y) \circ z = x \circ (y \circ
      z)</FMP>
</axiom>

<definition xml:id="semigroup.def">
  <CMP>
  A <term cd="semigroup" name="Semigroup" role="definiedum">
      semigroup</term>
   is a <term cd="semigroup" name="base-set" role="definiens">set S
       </term> with
   <term cd="semigroup" name="operation" role="definiens">binary
      operation \circ </term>
   : M x M -> M, obeying the following axioms:
   <axiom xml:href="closed.ax" />, <axiom xml:href="assoc.ax" />
  </CMP>
  <FMP logic="OntLang">
     <om:OMOBJ>
      <om:OMA>
        <om:OMS cd="OntLang" name="OConceptDef" />
        <om:OMS cd="semigroup" name="Semigroup" />
        <om:OMA>
          <om:OMS cd="OntLang" name="OIntersection" />
          <om:OMS cd="semigroup" name="AssociativeOperation" />
          <om:OMA>
            <om:OMS cd="OntLang" name="OExistsRelationWith" />
```

```
                <om:OMS cd="semigroup" name="base-set" />
                <om:OMS cd="[...]" name="Set" />
              </om:OMA>
            </om:OMA>
          </om:OMA>
        </om:OMOBJ>
      </FMP>
    </definition

</theory>
```

Listing 13: Definition of Semi-Group in OMDoc

4.3.5 Primitive or full Definition?

Due to the limitations of the logical formalism the meaning of some mathematical concepts cannot be captured in the Math Ontology Language. Chapter 2.4 investigates this shortcoming in more detail and gives with `AssociativeOperation` an example of a concept whose underlying semantics cannot be expressed in Description Logics. Consequently, some mathematical concepts must be introduced as atoms. The Math Ontology Language provides the construct `OConceptNecessaryDef` to define atomic concepts by constraining the meaning through necessary (but not sufficient) conditions. For example, the statement
`OConceptNecessaryDef {\DifferentiableFunction} {\Function}`
defines the atomic concept of a *differentiable function* and constrains its meaning in that every differentiable function must necessarily be a function.

Building on atomic concepts (and relations[46]) one can write down full definitions (giving necessary and sufficient conditions) for other mathematical concepts using the `OConceptDef` construct. For instance, assuming

- `SurjectiveFunction`
- `InjectiveFunction`

as atomic concepts, `BijectiveFunction` can be defined as:

```
\OConceptDef{\BijectiveFunction}
            {\OIntersection{\InjectiveFunction,\SurjectiveFunction}}.
```

Hence the question of when to fully define a mathematical concept and when to introduce it as an atomic concept (which is called a primitive definition) naturally arises.

One shall fully define a mathematical concept if and only if all its theory constitutive elements are included in the definition.

[46]Relations are quite similar to atomic concepts in that their meaning cannot be defined in DL but is implicitly captured in the name.

In the extensional approach to writing mathematical content, a concept is fully defined via the theory constitutive elements and hence the ontological definition is logically sound if and only if all these elements are included in the definition. The theory constitutive elements itself might be atomic.
To illustrate this best practice consider the definition of a ring given in formula 9. This definition does not include the distributivity axiom which is constitutive to the theory of rings and hence it is primitive. Formula 10 gives a revised version which includes distributivity and hence comprises a full definition.

Ring	$\sqsubseteq$	$\exists$base-set.NonEmptySet $\sqcap$ $\exists$additive-operation.(AssociativeOp $\sqcap$ IdentityOp $\sqcap$ InvertibleOp $\sqcap$ CommutativeOp) $\sqcap$ $\exists$multiplicative-operation.(AssociativeOp $\sqcap$ IdentityOp $\sqcap$ InvertibleOp)

Table 9: Definition of ring without distributitivity

Ring	$\equiv$	$\exists$base-set.NonEmptySet $\sqcap$ $\exists$additive-operation.(AssociativeOp $\sqcap$ IdentityOp $\sqcap$ InvertibleOp $\sqcap$ CommutativeOp) $\sqcap$ $\exists$multiplicative-operation.(AssociativeOp $\sqcap$ IdentityOp $\sqcap$ InvertibleOp) $\sqcap$ ($=_2$ distributive-operations)

Table 10: Definition of ring including distributivity

A closer look at definition 10 casts considerable doubt on the logical soundness of the statement. In fact, if we were to live up to the same standards as, for instance, theorem prover libraries, the definition of a ring would be incorrect, because it it does not state, for example, that the multiplicative operation is restricted to a subset of the base set of the ring (commonly referred to as R^*). But as we have argued repeatedly before, the Mathematical Semantic Web does not try to attain these standards, because it focuses on humans.

Thus, we advocate to redefine logical soundness in the context of the Mathematical Semantic Web as soundness up to common mathematical sense and implicitly shared understanding.
Of course this is a controversial proposition and might lead to inconsistencies in the knowledge base, but it greatly simplifies knowledge formalization and facilitates an intuitive modeling approach.

4.3.6 Concept or Instance?

In Description Logics concepts and instances are treated as two disjoint groups of knowledge objects. Yet, in knowledge modeling the border between the two is sometimes not so clear. It is hence important to understand the difference in order to make an informed choice on whether to use a concept or an instance in a particular situation.

A concept is a kind of thing, or in mathematical terms, a class of objects which share certain properties (somewhat similar to a generalized category from algebraic category theory). An instance is one particular object from such a concept. [47]

Despite this explicit distinction, one needs to consider the context in which a given mathematical object occurs to arrive at a conclusion on whether to introduce it as a concept or an instance. If it serves as an example for a more general concept, it must certainly be treated as an instance, whereas, if the object is understood as a handle for a mathematical theory, it most surely defines a concept.

To illustrate this distinction, we consider the example of complex numbers. In a high school mathematics course, complex numbers are introduced in their own theory and are probably defined similar to example 11, one therefore talks about the concept of a complex number (also note that the concept name is written in singular form).
In a college level algebra course, on the other hand, complex numbers are used to exemplify the concept of a field, hence they are treated as an instance and declared similar to statement 12.

$$\text{ComplexNumber} \quad := \exists \text{imaginary.RealNumber} \sqcap \exists \text{real.RealNumber}$$

Table 11: Definition of Complex Numbers

Field(ComplexNumbers)
base-set(ComplexNumbers, $\mathbb{C}$)
additive-operation(ComplexNumbers, $+$)
multiplicative-operation(ComplexNumbers, $*$)

Table 12: Complex Numbers as an instance

Once again, an instance cannot denote the same knowledge object as a concept in Description Logic. In fact, it is not even possible to declare a logical link between the two which could be exploited for reasoning.
To create a link between instances and concepts which ultimately denote the same mathematical object and thereby help humans navigate the ontology of mathematics, one could use the RDF `seeAlso` element. `seeAlso` is an annotation which is ignored by reasoning systems but used for navigation and supported by Semantic Web tools. Consequently, one might choose to extend the Math Ontology Language to provide support for links.

[47] Also, see [Noy and McGuinness, 2001, chapter 4.6] and [Nagypál, 2005, chapter 3] for a general discussion on this topic.

4.3.7 Concept or Relation?

In modeling mathematical knowledge in the Math Ontology Language one often times can use either atomic concepts or relations to define the mathematical concept at hand[48]. This choice is due to the limited expressiveness of the DL variant. Certain facts about mathematical objects cannot be written down explicitly in the logical formalism and hence must be captured implicitly via atomic concepts or relations (compare to chapter 2.4). Compare, for example, the two definitions of a *semigroup* given in the DL formulas 13 and 14, where the first makes use of relations and the second builds on atomic concepts. Both definitions are logically sound in that they capture the characteristics of a semigroup, but the first definition is superior, because it carries a richer machine accessible semantic and relates the concept `Semigroup` to previously defined concepts.

$$\text{Semigroup} := \exists \text{base-set.Set} \sqcap \exists \text{operation.AssociativeOperation}$$

Table 13: Definition of Semigroup using relations

$$\text{Semigroup} := \text{HasBaseSet} \sqcap \text{HasAssociativeOperation}$$

Table 14: Definition of Semigroup using atomic concepts

In general, the theory in which a particular concept occurs provides valuable information for deciding whether to use relations or atomic concepts. If the theory builds upon (i.e imports) other theories, the (atomic) concepts defined therein are like to occur in the definition of the concept at hand. To the contrary, the theory constitutive elements which relate the current to other theories shall be treated as relations. See 4.3.4 for an elaborate example.

4.3.8 Refine ontologies

One of the particularities of ontology engineering is that statements can be incrementally added to ontologies in order to monotonically extend the knowledge base. In practice, the author can start with primitive definitions of the concepts in her work (e.g. $\mathit{DifferentiableFunction} \sqsubseteq \mathit{Function}$) and later extend the ontology with more specific definitions (e.g. $\mathit{DifferentiableFunction} := \mathit{Function} \sqcap \exists \mathit{derivative.Function}$). Such extensions can be appended to an existing ontology and do not require any modifications of previous ontological statements.

We call such an extension to an ontology a **refinement** and encourage authors to make frequent use of refinements. The theory of refinements helps to overcome obstacles during knowledge modeling and reduces the likelihood of inconsistencies

[48] Also, see [Noy and McGuinness, 2001, chapter 4.5] and [Nagypál, 2005, chapter 3] for a general discussion on this topic.

in the knowledge base.

During knowledge modeling the author might reach a point where she is in doubt about the most accurate ontological statement. Being aware of the fact that refinements seamlessly extend the knowledge base, the author is willing to settle for a less precise definition in order to progress and to return to the issue later.

More importantly though, an author might be tempted to overshoot the goal of an accurate ontological definition by introducing statements that are too restrictive and cause inconsistencies in the knowledge base. The theory of refinements reduces the burden on the author to arrive at the most accurate definition at the first try which improves the quality of ontologies.

To illustrate this point we consider the concept `Function`. An author might introduce the following definition in his work:

$$\mathit{Function} \sqsubseteq \exists \mathit{domain}.\mathit{Set} \sqcap \exists \mathit{range}.\mathit{Set} \tag{8}$$

Having observed that every function has exactly one domain, he might furthermore state:

$$\mathit{Function}(\mathit{domain}) \tag{9}$$

These statements effectively overshoot the goal of an accurate definition, because the relation `domain` is also used in the general theory of n-ary relations, which means that the following definition of a *ternary relation* would cause an inconsistency in combination with the statements above:

$$\mathit{TernaryRelation} \sqsubseteq \mathit{Relation} \sqcap (=_3 \mathit{domain}) \tag{10}$$

This example furthermore motivates the general observation, that one has to be careful with assigning properties to relations. Properties of relations (e.g. *Function*, *InverseFunction*, *SymmetricRelation*, and *TransitiveRelation*) are global which means that they impose constraints on the relation which must be satisfied over the entire knowledge base. Careless use of relation properties can cause inconsistencies as in the example above. Consequently, relation properties shall only be used when they describe an intrinsic property of the relation. For example, the relation `equality` can safely be declared symmetric.
In the example above, being a function is not an intrinsic property of the relation `domain`, hence the following alternative to the statement above is suggested:

$$\mathit{Function} \sqsubseteq \exists \mathit{domain}.\mathit{Set} \sqcap \exists \mathit{range}.\mathit{Set} \sqcap (=_1 \mathit{domain}) \sqcap (=_1 \mathit{range}) \tag{11}$$

4.3.9 Definitions and Assertions

The Math Ontology Language provides with `OConceptDef` and `OConceptEquiv` (and similarly `OConceptNecessaryDef` and `OConceptSubsume`) two constructs which

have the same semantics in terms of their Description Logic counterparts. Nevertheless to ensure a high quality ontology of mathematics it is important to distinguish between the two. The `OConceptDef` is used for ontological statements which define the concept at hand and is therefore typically used in definitions. `OConceptEquiv`, on the other hand, formalizes assertions about mathematical concepts and is hence typically used in theorems, lemmas and the like.

Complex Analysis provides us with an example to illustrate this point. An *analytic function* is a complex-valued function which has a power series representation[49]. In addition, one can prove that *analytic functions* are equivalent to *holomorphic functions*. So, one uses `OConceptDef` to define *analytic functions* and `OConceptEquiv` to state an equivalence with *holomorphic functions*. The following example illustrates this point.

```
\begin{definition}[id=analyticFunction.def,for=AnalyticFunction]
  [...]
  \begin{FMP}
    \OConceptDef{\AnalyticFunction}
      {\OIntersection{\ComplexFunction,\OExistsRelationWith{\
          haspowerseries}{\PowerSeries}}}
  \end{FMP}
\end{definition}
  \begin{assertion}[type=theorem]
  [...]
  \begin{FMP}
    \OConceptEquiv{\AnalyticFunction}{\HolomorphicFunction}
  \end{FMP}
\end{assertion}
```

Listing 14: sTeX example to show the difference between definitions and assertions

[49] In this simple example we consider *analytic functions* as being analytic on the entire complex plane.

4.4 Heterogeneous Ontologies

On the Mathematical Semantic Web proposed in this work a heterogeneous group of mathematical authors dispersed over the entire planet develop an ontology of mathematics without central authority or coordination. In section 4.2.6 we concluded that neither a methodology nor a fixed set of rules can be imposed but rather suggest a collection of best practices in 4.3.
Due to this setup one might prognosticate that the Mathematical Semantic Web will degenerate to a pool of unconnected ontologies instead of the enormous networked mathematical knowledge base we envision. The following two reasons suggest that our prediction is more likely to materialize:

- **Ontology reuse**
  Reusing ontologies is a fundamental prerequisite for the success of the Mathematical Semantic Web as argued in 4.2.5 and 3.3.2, because it creates links between ontologies and ultimately ties them together into one knowledge base.
  It is in the interest of an author to reuse existing ontologies, because it reduces the annotation effort and adds value to her work. Hence the degree of ontology reuse hinges on the quality of tool support for discovering relevant ontologies and for integrating those into the current work. Chapter 3.3.2 provides an initial analysis of this topic, but due to its significance further research is necessary.

- **Shared conceptual understanding**
  In section 4.2.2 we argued that mathematical discourse is semi-formal and carries an inherent conceptualization.
  We extend the argument in that mathematicians from one Community of Practice (see 4.4.1) share an understanding for mathematical objects and their characteristics. This understanding is developed during education in mathematics and facilitates communication among mathematicians. Due to this fact it is reasonable to assume that mathematicians from one Community of Practice naturally agree on a formal conceptualization which means that ontologies developed by different authors are compatible and streamlined.

4.4.1 Communities of Practice

Communities of Practice are an important concept in order to understand how loosely bound mathematician can still collaborate in the development of a shared ontology in line with the argument made above.

Following the definition in [Kohlhase and Kohlhase, 2005], a **Community of Practice** (CoP) is "a group of people (e.g. professionals) informally bound to one another through exposure to a common class of problems, common pursuit of

solutions, and thereby themselves embodying a store of knowledge."
Applying this definition to professionals in the field of mathematics one can deduce that members of one CoP in mathematics share a conceptualization of the mathematical realm in which they operate.

On the other hand, the definition also implies that different CoPs differ in their conceptualization of mathematics. A different conceptualization can reflect itself in the definition of concepts. The concept of a *ring* from algebra provides an illustrative example. In some CoPs a ring is defined as an additive abelian group together with a multiplicative semigroup obeying the distributive laws, whereas other CoPs define a ring to have a monoid as the multiplicative structure. Hence the two conceptualizations differ in the existence of a multiplicative identity.

Chapter 3.3.2 discusses tool support for the Mathematical Semantic Web and suggests that editors should provide functionality to discover relevant, existing ontologies. Communities of Practice help to understand the meaning of the word "relevant" in this context. Ontologies are relevant to the mathematical author if they relate to the current work (i.e. contain similar or related concepts) and if they were developed within the CoP of which the author is a member. Hence, advanced editor need to model and monitor CoPs[50].

To arrive at one global knowledge base for mathematics assembled from the ontologies developed within the various Communities of Practice, mappings between the ontologies have to be created. One option is to rely on humans to annotate equivalences between concepts, relations and instances. In line with the example above, the concept `Ring` from CoP A would be declared equivalent to the concept `RingWithoutIdentity` from CoP B.
Automatically deriving such mappings is an open research question in the context of the Semantic Web and one could investigate whether limiting the application domain to mathematics reduces the complexity of the problem.

In addition, the Communities of Practice must be considered when developing incentive structures targeted at mathematical authors in order to motivate contributions to the Mathematical Semantic Web.

Although Communities of Practice are an important aspect of the Mathematical Semantic Web, a detailed analysis is beyond the scope of this work and must be subject to future research.

[50]Tool support for Communities of Practice is the subject of research conducted as part of the Lectora project at Jacobs University Bremen.

5 Future Work

The work presented in this paper must be understood as a theoretical exposition based on previous results and independent case studies. Ultimately, the prospects of a Mathematical Semantic Web and the vast knowledge base of mathematics we envision can only be determined in a large field study involving mathematicians pursuing their daily business. Our work provides a thorough analysis which justifies the engagement in such a comprehensive and time-intensive experiment.

Before one can conduct such a large experiment, the infrastructure and architecture of the Mathematical Semantic Web as described in chapter 3 must be fully implemented. At this point, only the OMDoc and sTEX components are functional which leads to the following requirements list:

- **Authoring Environment**
  Authoring environments for the LaTeX typesetting program need to be extended by functionalities which support the author in appending ontological statements to her documents and in discovering existing ontologies. The chapter on authoring environment evaluation (3.3.2) is dedicated entirely to this topic and suggestions made there should be implemented in order of feasibility and importance.
  So far, a parser for sTEX documents and the ontological statements contained therein has been implemented which validates the correctness of formulas in the Math Ontology Language and alerts the user to errors and warnings.
  In addition to conventional authoring environments for mathematical documents one could adapt SWiM [Lange, 2007] to serve as a front end for the Mathematical Semantic Web. SWiM is a semantic wiki for collaboratively building, editing and browsing a mathematical knowledge base.

- **Document Conversion**
  Figure 6 illustrates a number of automatic conversion steps which take place when an author publishes his sTEX document on the Mathematical Semantic Web. Although these conversions are semi-trivial (i.e. XSLT stylesheets) or use existing tools (i.e. LaTeXML), none of them has been fully implemented. Furthermore the process of publishing the resulting files (i.e. OMDoc and OWL) on the Mathematical Semantic Web has yet to be specified and might require the installation of a central repository, similar to arXiv.org.

- **Reasoning on Structural and Domain Knowledge**
  In chapter 3.5 we suggest a way to combine structural and domain knowledge in the reasoning process. This proposal would benefit from additional theoretical deliberation and the implementation of a prototype to demonstrate the feasibility of the two step reasoning process.

- **Services**
  To prove the usefulness of the Mathematical Semantic Web to its contributors, the services described in chapter 3.6 shall be implemented.

In addition to the implementation, the theoretical underpinning of the Mathematical Semantic Web, which is initiated with this document, needs to be supplemented in order to gain a complete understanding of the project. Further research should be conducted on the following topics:

- **Math Ontology Language**
  The experience gained during a large field study will help to analyse the design choices made for the Math Ontology Language (see section 3.2.2) and might suggest modifications.

- **Extend Best Practices**
  Similarly, during the annotation of a large corpus of mathematical documents the Best Practices from chapter 4.3 shall be applied and evaluated which leads to modifications and additional Best Practices.

- **Community of Practice**
  Due to the scope of this work we could only briefly touch upon the topic of Communities of Practice (see section 4.4.1) which is nevertheless important to the entire project and must be subject of further research in order to identify useful services and derive an incentive structure for mathematicians. Since Communities of Practice are central to the Lectora project [Müller,] synergies can be leveraged by close collaboration between the two projects.

Given the enormous scope of the project one could fill entire pages with implementation requirements, theory gaps, possible extensions and improvements. We restrict ourselves to the list above for a start and acknowledge that it is not complete.

6 Conclusion

The initial survey on the utility of Description Logics for formalizing mathematical knowledge concluded with the proposal of a Mathematical Semantic Web. The Mathematical Semantic Web we envision is a enormous, distributed, and dynamic knowledge base aggregated from individual ontologies published all over the Internet, together with service offerings which operate on the ontologies to provide intelligent functionality to the math worker.

We have introduced the architecture, individual building blocks and overall infrastructure of the Mathematical Semantic Web in this paper. Our work provides the theoretical underpinning and guide for future implementation (see chapter 5) of the extension of the Semantic Web we propose.
Irrespective of the nature of its foundations, the success of the Mathematical Semantic Web will ultimately depend on whether a majority of authors of mathematical documents perceive that its benefits to the individual are worth more than the time invested into adding semantic annotations.

To highlight the benefits of the Mathematical Semantic Web we described a number of valuable and unique services for mathematicians, the most prominent one being an advanced search engine. The range and number of services possible is only constrained by our imagination and the list given in this paper is by no means exhaustive.
Furthermore, we introduced a method to combine domain and structural knowledge in reasoning processes, thereby leveraging even more of the knowledge contained in mathematical documents at no additional cost.

A significant fraction of our research aimed at lowering the entry barrier to semantically annotating mathematical documents and to ease the knowledge modeling process in Description Logics for mathematics authors in general. We effectively reduced the time investment for semantic annotations in the context of the Mathematical Semantic Web through the following measures:

- Development of a Description Logic variant for the mathematical realm which can be readily understood by mathematicians with a background in set theory.
- Integration of ontological statements into the original mathematical document using the OMDoc format as a container for both.
- Automation of the publication process for the Mathematical Semantic Web.
- Invasion of LaTeX authoring environments through an extension of the sTeX macro packages.

- Suggestion of Best Practices for semantic annotations in the Math Ontology Language.

For the reasons summarized above it is safe to assume that the Mathematical Semantic Web will be widely accepted in the mathematical community which justifies further research on the topic and an implementation guided by the architectural suggestions made in this exposition.

A Appendix

A.1 Math Ontology Language extension for sTEX

```
\begin{module}[id=OntLang]

  % ---------- Global concept definitions
     ------------------------------

  \symdef{OTopConcept}{\top}
  \symdef{OBottomConcept}{\bot}

  %---------- Concept constructors
     ------------------------------------

  \symdef{OConceptDef}[2]{{#1} := {#2}}
  \symdef{OConceptNecessaryDef}[2]{{#1} \ddot{\sqsubseteq} {#2}}
  \symdef{OConceptEquiv}[2]{{#1} \equiv {#2}}
  \symdef{OConceptSubsume}[2]{{#1} \sqsubseteq {#2}}

  \symdef{OUnion}[1]{\assoc[p=500]{\sqcup}{#1}}
  \symdef{OIntersection}[1]{\assoc[p=500]{\sqcap}{#1}}
  \symdef{OComplement}[1]{\neg {#1}}

  \symdef{ODisjoint}[2]{{#1} \sqcap {#2} = \OBottomConcept}

  %---------- Relation Constructors
     ------------------------------------

  \symdef{ORelationDef}[2]{{#1} =_R {#2}}
  \symdef{ORelationNecessaryDef}[2]{{#1} \ddot{\sqsubseteq}_R {#2}}
  \symdef{ORelationEquiv}[2]{{#1} \equiv_R {#2}}
  \symdef{ORelationSubsume}[2]{{#1} \sqsubseteq_R {#2}}

  \symdef{ODomain}[2]{\mathbb{D}({#1}) = {#2}}
  \symdef{ORange}[2]{\mathbb{R}({#1}) = {#2}}

  \symdef{OInverse}[2]{{#1}^{-1} = {#2}}
  \symdef{OFunction}[1]{{#1} \in Functions}
  \symdef{OInverseFunction}[1]{{#1}^{-1} \in Functions}
  \symdef{OTransitiveRel}[1]{{#1} \in TransitiveRelations}
  \symdef{OSymmetricRel}[1]{{#1} \in SymmetricRelations}

  %---------- Concept constructors using relations
     ---------------------

  \symdef{ORestrictRelationTo}[2]{\forall{#1}\sdot{#2}}
  \symdef{OExistsRelationWith}[2]{\exists{#1}\sdot{#2}}

  \symdef{OMaxCardinality}[2]{{#1} \leq_{\#} {#2}}
  \symdef{OMinCardinality}[2]{{#1} \geq_{\#} {#2}}
  \symdef{OExcatCardinality}[2]{{#1} =_{\#} {#2}}
```

```
  % ---------- Instance constructors
     ----------------------------------

47  \symdef{OInstanceOf}[2]{{#1}:{#2}}
    \symdef{OAnonymInstanceOf}[1]{?:{#1}}
    \symdef{ORelatedTo}[2]{{#1} \leadsto {#2}}
    \symdef{OIdenticalTo}[1]{= {#1}}
    \symdef{ODifferentFrom}[1]{\neq {#1}}
52  \symdef{OInstanceDef}[3]{{#1}:{#2} \mid \assoc[p=500]{\|}{#3}}
    \symdef{OAnonymInstanceDef}[2]{?:{#1} \mid \assoc[p=500]{\|}{#2}}

\end{module}
```

Listing 15: Definition of the Math Ontology Language in sTeX

A.2 Math Realm Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [ <!--Entity definitions --> ]>

<rdf:RDF
 5  <!-- Namespace definitions --> >
      <owl:Ontology rdf:about="">
        <rdfs:comment>The base math structure ontology</rdfs:comment>
      <rdfs:label>Base Math Structure Ontology</rdfs:label>
     <!-- Further statements -->
10  </owl:Ontology>

      <owl:Class rdf:about="MathRealm">
      <rdfs:subClassOf rdf:resource="&owl;Thing" />
    </owl:Class>
15
    <owl:Class rdf:about="MathConcept">
      <rdfs:subClassOf>
        <owl:Class rdf:about="MathRealm" />
      </rdfs:subClassOf>
20  </owl:Class>
    <owl:Class rdf:about="MathRelation">
      <rdfs:subClassOf>
        <owl:Class rdf:about="MathRealm" />
      </rdfs:subClassOf>
25    <owl:disjointWith rdf:resource="MathConcept" />
    </owl:Class>
    <owl:Class rdf:about="MathObject">
      <rdfs:subClassOf>
        <owl:Class rdf:about="MathRealm" />
30    </rdfs:subClassOf>
      <owl:disjointWith rdf:resource="MathConcept" />
      <owl:disjointWith rdf:resource="MathRelation" />
    </owl:Class>
```

```
  <owl:ObjectProperty rdf:about="declaredIn">
    <rdfs:domain rdf:resource="MathRealm"/>
    <rdfs:range rdf:resource="&omdoc;OMDocConcept"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="definedIn">
    <rdfs:domain rdf:resource="MathRealm"/>
    <rdfs:range rdf:resource="&omdoc;OMDocConcept"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="occursIn">
    <rdfs:domain rdf:resource="MathRealm"/>
    <rdfs:range rdf:resource="&omdoc;OMDocConcept"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

Listing 16: Math Realm Ontology

References

[Sem,] Semantic web layered cake, retrieved from http://www.w3c.rl.ac.uk/pasttalks/slidemaker/km_europe/slide5-2.html on may 3rd, 2007.

[zen,] Zentralblatt math, facts and figures, retrieved from http://www.zblmath.fiz-karlsruhe.de/math/about/zentralblatt on may 3rd, 2007.

[Aboul-Hosn, 2006] Aboul-Hosn, K. (2006). *A Proof-Theoretic Approach to Mathematical Knowledge Management*. PhD thesis, Cornell University.

[Akrivi et al., 2006] Akrivi, K., Elena, T., Constantin, H., Georgios, L., and Costas, V. (2006). A comparative study of four ontology visualization techniques in protege: Experiment setup and preliminary results. In *Proceedings of the conference on Information Visualization*.

[Andréka et al., 1996] Andréka, H., van Benthem, J., and Németi, I. (1996). Modal languages and bounded fragments of predicate logic. ILLC ML-96-03, University of Amsterdam.

[Antoniou and VanHarmelen, 2004] Antoniou, G. and VanHarmelen, F. (2004). *A Semantic Web Primer*. B&T.

[Auer, 2006] Auer, S. (2006). Rapidowl - an agile knowledge engineering methodology. In *Proceedings of the Sixth International Andrei Ershov Memorial Conference "Perspectives of System Informatics"*.

[Baader et al., 2003] Baader, F., Calvanese, D., and McGuinness, D. (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 1. edition edition. ISBN-10: 0521781760 ISBN-13: 978-0521781763.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*.

[Bourbaki, 1998] Bourbaki, N. (1998). *Elements of Mathematics. Algebra I. Chapters 1 - 3*. Springer, Berlin.

[Caprotti et al., 2004] Caprotti, O., Dewar, M., Turi, D., Asperti, A., Bancerek, G., and Trybulee, A. (2004). Mathematical service matching using description logic and owl. *Lecture notes in computer science*, 3119:73–87.

[Ding et al., 2005] Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., and Kolari, P. (2005). Finding and ranking knowledge on the semantic web. In *Proceedings of the 4th International Semantic Web Conference*.

[Fernández-López and Gómez-Pérez, 2002] Fernández-López, M. and Gómez-Pérez, A. (2002). Deliverable 1.4: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. Technical report, EU IST Project IST-2000-29243 OntoWeb.

[Fürst et al., 2001] Fürst, F., Leclère, M., and Trichet, F. (2001). Contribution of the ontology engineering to mathematical knowledge management. In *1st Int. Workshop on Mathematical Knowledge Management*.

[Fürst et al., 2003] Fürst, F., Leclère, M., and Trichet, F. (2003). Ontological engineering and mathematical knowledge management: A formalization of projective geometry. *Annals of Mathematics and Artificial Intelligence*, 38:65–89.

[Genesereth and Nilsson, 1987] Genesereth, R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA. ISBN: 978-0934613316.

[Gruber and Olsen, 1994] Gruber, T. and Olsen, G. (1994). An ontology for engineering mathematics. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning*.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

[Hungerford, 1974] Hungerford, T. W. (1974). *Algebra*. Springer Science+Business Media, Inc.

[Knublauch et al., 2004] Knublauch, H., Musen, M., and Rector, A. (2004). Editing description logic ontologies with the protégé owl plugin. In *International Workshop on Description Logics*.

[Kohlhase and Kohlhase, 2005] Kohlhase, A. and Kohlhase, M. (2005). An exploration in the space of mathematical knowledge. In *Mathematical Knowledge Management*.

[Kohlhase, 2004] Kohlhase, M. (2004). Semantic markup for TEX / LATEX. In *Proc. of the Mathematical User-Interface Workshop at MKM'04, Bialowieza, Poland*.

[Kohlhase, 2006] Kohlhase, M. (2006). *OMDoc. An Open Markup Format for Mathematical Documents [version 1.2]*. Springer-Verlag GmbH.

[Kohlhase and Rabe,] Kohlhase, M. and Rabe, F. Theory morphisms as first-class objects in omdoc. To be published - just borrowed some formulations.

[Lange, 2007] Lange, C. (2007). Swim - a semantic wiki for mathematical knowledge management. Technical report, Jacobs University Bremen.

[Mahnke and Scheffczyk, 2005] Mahnke, A. and Scheffczyk, J. (2005). Engineering mathematical knowledge. In *Mathematical Knowledge Managemenet, MKM 2005*, pages 250–266.

[Miller,] Miller, B. *LaTeXML: A LaTeX to XML converter. Web Manual at http://dlmf.nist.gov/LaTeXML.* National Institute of Standards and Technology.

[Müller,] Müller, C. *Lectora: An interactive and collaborative reader for scientific documents. Web Manual at http://kwarc.info/projects/lectora/index.html.* Jacobs University Bremen.

[Nagypál, 2005] Nagypál, G. (2005). Methodology for building sws ontologies in dip.

[Noy and McGuinness, 2001] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University.

[Sattler,] Sattler, U. List of dl reasoners at: http://www.cs.man.ac.uk/ sattler/reasoners.html.

[Shadbolt et al., 2006] Shadbolt, N., Hall, W., and Berners-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent systems*, 21:96–101.

[Society, 2000] Society, A. M. (2000). Mathematics subject classification 2000.

[Sommerville, 2004] Sommerville, I. (2004). *Software Engineering 7*. Pearson Education Limited.

[Storey et al., 2004] Storey, M., Lintern, R., Ernst, N., and Perrin, D. (2004). Visualization and protégé.

[Witbrock et al., 2004] Witbrock, M., Panton, K., Reed, S. L., Schneider, D., Aldag, B., Reimers, M., and Bertolo, S. (2004). Automated owl annotation assisted by large knowledge base.

Matthias Bröcheler is a PhD student in the Department of Computer Science at the University of Maryland, College Park. His research interests include knowledge representation, information extraction, information integration, and data mining. He received his Bachelor of Science in Mathematics and Computer Science from Jacobs University in Bremen, Germany. Contact him via email at mbroechele@jacobs-alumni.de.