An Open Markup Format for Mathematical Documents OMDoc [Version 1.6 (pre-2.0)]

December 14, 2015

Abstract:The OMDoc (Open Mathematical Documents) format is a content markup scheme for (collections of) mathematical documents including articles, textbooks, interactive books, and courses. OMDoc also serves as the content language for agent communication of mathematical services on a mathematical software bus.

This document is the specification of Version 1.6 of OMDoc of the OMDoc format, the first step towards OMDoc2. It defines the OMDoc language features and their meaning. The content of this part is normative for the OMDoc format; an OMDoc document is valid as an OMDoc document, iff it meets all the constraints imposed here. OMDoc applications will normally presuppose valid OMDoc documents and only exhibit the intended behavior on such.

Contents

Preface 3

I The OMDoc Format 5

1 Dimensions of Representation in OMDoc 7

2 OMDoc as a Modular Format 11

3 The OMDoc Namespaces 13

4 Common Attributes in OMDoc 15

Preface

The OMDoc (Open Mathematical Documents) format is a content markup scheme for (collections of) mathematical documents including articles, textbooks, interactive books, and courses. OMDoc also serves as the content language for agent communication of mathematical services on a mathematical software bus.

This document is the specification of Version 1.6 of OMDoc of the OMDoc format, the first step towards OMDoc2. It defines the OMDoc language features and their meaning. The content of this part is normative for the OMDoc format; an OMDoc document is valid as an OMDoc document, iff it meets all the constraints imposed here. OMDoc applications will normally presuppose valid OMDoc documents and only exhibit the intended behavior on such.

Part I

The OMDoc Format

In this chapter we will discuss issues that pertain to the general setup of the OMDoc format, before we present the respective modules in later chapters. OMDoc1.6 is the first step towards a second version of the OMDoc format.

Chapter 1

Dimensions of Representation in OMDoc

BNP:1

Strict vs. Pragmatic The OMDoc format is divided into two sub-languages: "Strict" OMDoc (in the lower half of Figure 1.1) and "Pragmatic" OMDoc (in the upper half[2]). EdN:2 The first subset uses a minimal set of elements representing the meaning of a mathematical expression in a uniform structure, while the second one tries to strike a pragmatic balance between verbosity and formality. Both forms of content expressions are legitimate and have their role in representing mathematics. The strict OMDoc format features a minimal set of conceptually orthogonal representational primitives, resulting in expressions with canonical structure, which simplifies the implementation of OMDoc processors as well as the comparison of content expressions. The pragmatic OMDoc format provides a large representational infrastructure that aims at being intuitive for humans to understand, read, and write.[3] EdN:3 In particular, the simplicity and conceptual clarity of strict OMDoc make allow to express structural well-formedness constraints, whereas the vocabulary of pragmatic OMDocis much nearer to mathematical practice and is thus easier to learn. It is a crucial design choice of the OMDocformat that the meaning fo pragmatic representations is defined entirely interms of strict representations[1]. Note that there may be multiple "pragmatic vocabularies" defined in terms of the strict core catering to different communities and their tastes.

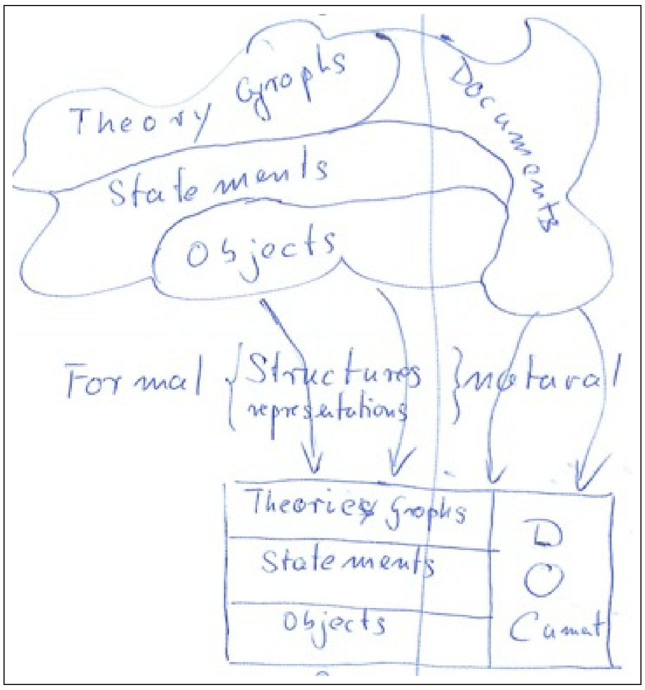

Figure 1.1: Dimensions of Representation in OMDoc

The introduction of strict OMDoc and the re-interpretation of pragmatic OMDoc in terms of it

[1]New Part: re-read and strengthen the argumentation

[2]EdNote: add the words "strict" and "pragmatic" to the picture

[3]EdNote: maybe state the numbers of elements in the end

[1]The strategy of dividing a markup format into a simple and structurally elegant core language and a larger set of pragmatic extensions which can be given a meaning by translating into the core was first pioneered by the author for content MathML3 [**CarlisleEd:MathML08**]

is radical redesign of the OMDoc format, which is new in OMDoc1.6. For this reason we consider OMDoc1.6 the first step into the direction of OMDoc2. With the development of strict OMDoc we aim to identify the representational primitives for representing mathematical documents, which can be given a simple and elegant semantics.

Formal vs. Informal One of the hallmarks of mathematical language is that is very rigorous in structure and usage in an attempt to fix the meaning of (mathematical) objects and statements about them. Indeed, the first decades of the last century established that mathematical language can in principle be expanded into logical form, where all objects and statements are fully identified by their syntactic form, and all reasoning steps are similarly justified by their form alone. we speak of "formal mathematics", when this is exercised and of "formal reasoning", when proofs are carried out in logical systems on this basis . In the last decades, significant parts of mathematical knowledge have been formalized and verified with the help of computers. But formalization and formal reasoning is still so costly and tedious that only a very small part of mathematics is formalized and verified in practice. Currently almost all mathematical documents consist of a mix of formal and informal (i.e. natural language) elements — certainly during the development of mathematical knowledge, but also in publications. Therefore are representation format for mathematical documents must allow this as well, consequently, OMDochas two sub-languages, "formal OMDoc" (on the left side of Figure 1.1) and "natural OMDoc" (on the right side).

OMDocoffers markup at three levels: objects, statements, and context.

objects are usually represented as *formulae* or *natural language phrases* in mathematical documents. In formal OMDocformulae are marked up according to their functional structure (as operator trees) and according to their layout in informal OMDoc(as layout trees). Note that any object can be represented in all three ways and all three ways of representation can be mixed at any level to account for mathematical practice, e.g. for mixed formulae like $\{n \in \mathbb{N} \big| n > 3 \text{ is prime}\}$.

statements are usually represented as *natural language sentences (with formulae)*[2] in informal settings and as (logical) formulae in formal ones. The discussion about the three ways of representation of objects applies analogously. Note that functional markup in formal OMDoconly addresses part of the requirements of formality, since their meaning depends on their context; we will explore this next.

theory graphs The context of objects (and the statements that contain them) is given by special statements (declarations). For conciseness and tractability, OMDocgroups declarations into "theories" and connects them by "theory morphisms" into "theory graphs". In a nutshell, every object (and thus every statement) has a "home theory", in which is meaningful. Theory morphisms make objects and statements available in their target theories.

As statements, theories and theory graphs are large objects, their informal representations (as mathematical text fragments and documents) usually carry linguistic cues to their discourse structure[4]. We discuss the relation between the discourse structure of informal representations and the formal structure of statements and theory graphs next. EdN:4

Discourse vs. Content Structure Mathematical Documents are very explicitly structured to help the reader grasp the complex objects, their relationships, and the flow of the argumentation in the proofs: Objects are often represented as formulae that reveal their structure, statements are labeled by indicators to their epistemic contribution to context (e.g. by labeling them as "definitions" or "theorems") and numbered for exact reference. The exposition of larger documents usually follows a topical structure with superimposed narrative structure driven by knowledge dependencies rather than e.g. a temporal dramaturgy driven by suspense. Even so, the structure of an informal document may be quite different from the formal structure of the knowledge it introduces. For instance, when we introduce a new concept in a course, we often first introduce a naive reduced approximation $\mathcal{N}$ of the real theory $\mathcal{F}$, only to show an example $\mathcal{E}_{\mathcal{N}}$ of where this

[2]or even larger text fragments made up of sentences like paragraphs

[4]EdNote: change the "documents" in Figure 1.1 to "discourse", at least in the strict box

is insufficient. Then we propose a first (straw-man) solution $\mathcal{S}$, and show an example $\mathcal{E}_\mathcal{S}$ of why this does not work. Based on the information we gleaned from this failed attempt, we build the eventual version $\mathcal{F}$ of the concept or theory and demonstrate that this works on $\mathcal{E}_\mathcal{F}$.

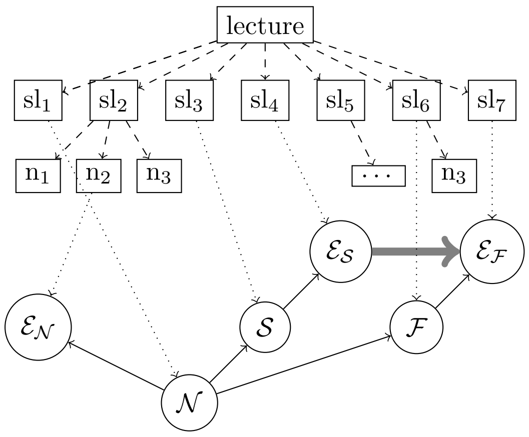

Figure 1.2: Content vs. Narrative Structures

The structure with the solid lines and boxes at the bottom of Figure 1.2 represents the content structure, where the circles $\mathcal{N}$, $\mathcal{E}_\mathcal{N}$, $\mathcal{S}$, $\mathcal{E}_\mathcal{S}$, $\mathcal{F}$, and $\mathcal{E}_\mathcal{F}$ signify theories for the content of the respective concepts and examples. The arrows represent the theory inheritance structure, e.g. Theory $\mathcal{F}$ imports theory $\mathcal{N}$. The top part of the diagram with the dashed lines stands for the narrative structure, where the arrows mark up the document structure. For instance, the slides sl_i are grouped into a lecture. In the example in Figure 1.2, the second slide of "lecture" presents the first example: the text fragment n_1 introduces it, and n_2 presents $\mathcal{E}_\mathcal{N}$ and n_2 might say something like "this did not work in the current situation, so we have to extend the conceptualization…". In a conventional setting, the narrative structure on the top and the content structure would be represented in different documents: The lecture slides and the formalization, and the equivalences (e.g. that n_2 verbalizes $\mathcal{E}_\mathcal{N}$; we have visualized these relations as dotted arrows in Figure 1.2) could not be taken advantage of, since they are not explicitly represented.

But these equivalences can be utilized to render services to the reader, for instance the imports relation in the theory graph on the lower half of Figure 1.2 induces a dependency relation that can be used to generate a minimal explanation (without the motivation) of $\mathcal{E}_\mathcal{F}$. For an example at the object level, consider for instance the formula $a(x+y^2)$, whose layout is ambiguous in two places: a could be a factor in a product (presented as juxtaposition) or a function that is applied to an argument. Likewise y^2 could be the variable y raised to the second power or the second element in the sequence $y^1, y^2, \ldots, y^n$. Humans can usually disambiguate this from the context, but a screen reader service needs access to the operator tree to read this as "a times [pause] x plus y squared" or "a applied to [pause] x plus y two".

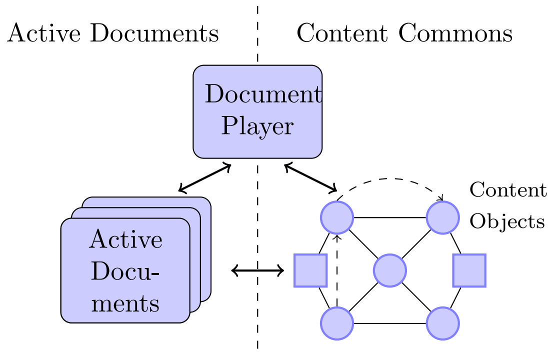

Figure 1.3: The Active Documents Paradigm

OMDocaims to reconcile the dichotomy between discourse structures (in informal mathematical documents which currently carry most of mathematical knowledge) and formal structures (that machines can operate upon) in one joint format. The central technique employed in OMDocis that of "parallel markup": The technique comes from MathML, where the `semantics` element is used to accomodate equivalent layout (presentation MathML) and operator trees (content MathML) and possibly foreign representations. Equivalence of nested sub-structures are represented by special cross-references. The MathML processor choses the one most adequate to its task — in the absence of distinguisthing information the first child.

OMDocextends this to the document level: The document contains elements whose children are alternative representations of the same object/statement/theory.[5] The significance of this is EdN:5
for that is Figure 1.3 shows the [6]. EdN:6

Just as for content-based systems on the formula level, there are now MKM systems that

[5]EdNote: implement this, and think about the cross-referencing, also need continuations to break tree overlaps, e.g.

generate presentation markup from content markup, based on general presentation principles, also on this level. For instance, the ACTIVEMATH system [**MelBue:krma03**] generates a simple narrative structure (the presentation; called a personalized book) from the underlying content structure (given in OMDoc) and a user model.

Coverage Currently our understanding of these primitives is largely limited to formal parts of mathematics, therefore strict OMDOC1.6 covers significantly less of informal mathematical documents than OMDOC1.2, so the meaning-giving translation from pragmatic OMDoc elements to strict OMDoc is partial. We plan to develop strict OMDoc into a system with greater coverage in the upcoming versions of OMDoc. OMDOC2.0 will be the first stable version where the coverage of strict OMDoc is complete. ENP:1

in content objects straddling slides.

[6]EDNOTE: talk about parallel markup, content documents and narrative documents and how to crosslink them and share structure

Chapter 2

OMDoc as a Modular Format

A modular approach to design is generally accepted as best practice in the development of any type of complex application. It separates the application's functionality into a number of "building blocks" or "modules", which are subsequently combined according to specific rules to form the entire application. This approach offers numerous advantages: The increased conceptual clarity allows developers to share ideas and code, and it encourages reuse by creating well-defined modules that perform a particular task. Modularization also reduces complexity by decomposition of the application's functionality and thus decreases debugging time by localizing errors due to design changes. Finally, flexibility and maintainability of the application are increased because single modules can be upgraded or replaced independently of others.

The OMDoc vocabulary has been split by thematic role, which we will briefly overview in Figure 2.1 before we go into the specifics of the respective modules in ?spec@mobj? to ?spec@quiz?. To avoid repetition, we will introduce some attributes already in this chapter that are shared by elements from all modules. In ?spec@document-model? we will discuss the OMDoc document model and possible sub-languages of OMDoc that only make use of parts of the functionality (?spec@sub-languages?).

The first four modules in Figure 2.1 are required (mathematical documents without them do not really make sense), the other ones are optional. The document-structuring elements in module DOC have an attribute `modules` that allows to specify which of the modules are used in a particular document (see ?spec@omdoc-infrastructure? and ?spec@sub-languages?).

Module	Title	Required?	Chapter
MOBJ	Mathematical Objects	yes	?spec@mobj?
Formulae are a central part of mathematical documents; this module integrates the content-oriented representation formats OpenMath *and* MathML *into OMDoc*			
MTXT	Mathematical Text	yes	?spec@mtext?
Mathematical vernacular, i.e. natural language with embedded formulae			
DOC	Document Infrastructure	yes	?spec@omdoc-infrastructure?
A basic infrastructure for assembling pieces of mathematical knowledge into functional documents and referencing their parts			
DC	Metadata	yes	?spec@dc-elements? and ?spec@dc-roles?
Contains bibliographical and licensing metadata ("data about data") which can be used to annotate many OMDoc elements by descriptive and administrative information that facilitates navigation and organization			
RT	Rich Text Structure	no	?spec@rt?
Rich text structure in mathematical vernacular (lists, paragraphs, tables, . . .)			
ST	Mathematical Statements	no	?spec@statements?
Markup for mathematical forms like theorems, axioms, definitions, and examples that can be used to specify or define properties of given mathematical objects and theories to group mathematical statements and provide a notion of context.			
PF	Proofs and proof objects	no	?spec@proofs?
Structure of proofs and argumentations at various levels of details and formality			
ADT	Abstract Data Types	no	?spec@adt?
Definition schemata for sets that are built up inductively from constructor symbols			
CTH	Complex Theories	no	?spec@complex-theories?
Theory morphisms; they can be used to structure mathematical theories			
DG	Development Graphs	no	?spec@development-graphs?
Infrastructure for managing theory inclusions, change management			
EXT	Applets, Code, and Data	no	?spec@ext?
Markup for applets, program code, and data (e.g. images, measurements, . . .)			
PRES	Presentation Information	no	?spec@pres?
Limited functionality for specifying presentation and notation information for local typographic conventions that cannot be determined by general principles alone			
QUIZ	Infrastructure for Assessments	no	?spec@quiz?
Markup for exercises integrated into the OMDoc document model			

Figure 2.1: The OMDoc Modules

Chapter 3

The OMDoc Namespaces

The namespace for the OMDoc2 format is the URI `http://omdoc.org/ns`. Note that the OMDoc namespace does not reflect the versions[1], this is done in the `version` attribute on the document root element `omdoc` (see ?spec@omdoc-infrastructure?). As a consequence, the OMDoc vocabulary identified by this namespace is not static, it can change with each new OMDoc version. However, if it does, the changes will be documented in later versions of the specification: the latest released version can be found at [**URL:omdocspec**].

In an OMDoc document, the OMDoc namespace must be specified either using a namespace declaration of the form `xmlns="http://omdoc.org/ns"` on the `omdoc` element or by prefixing the local names of the OMDoc elements by a namespace prefix (OMDoc customarily use the prefixes `omdoc:` or `o:`) that is declared by a namespace prefix declaration of the form `xmlns:o="http://omdoc.org/ns"` on some element dominating the OMDoc element in question (see for an introduction). OMDoc also uses the following namespaces[2]:

Format	namespace URI	see
Dublin Core	`http://purl.org/dc/elements/1.1/`	?spec@dc-elements? and ?spec@dc-roles?
Creative Commons	`http://creativecommons.org/ns`	?spec@creativecommons?
MathML	`http://www.w3.org/1998/Math/MathML`	?spec@cmml?
OpenMath	`http://www.openmath.org/OpenMath`	?spec@openmath?
XSLT	`http://www.w3.org/1999/XSL/Transform`	?spec@pres?

Thus a typical document root of an OMDoc document looks as follows:

```
<?xml version="1.0" encoding="utf−8"?>
<omdoc xml:id="test.omdoc" version="1.6"
  xmlns="http://omdoc.org/ns"
  xmlns:cc="http://creativecommons.org/ns"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:om="http://www.openmath.org/OpenMath"
  xmlns:m="http://www.w3.org/1998/Math/MathML">
...
</omdoc>
```

[1]The namespace is different from the OMDoc1 formats (versions 1.0, 1.1, and 1.2), which was `http://www.mathweb.org/omdoc`, but the OMDoc2 namespace will stay constant over all versions of the OMDoc2 format.

[2]In this specification we will use the namespace prefixes above on all the elements we reference in text unless they are in the OMDoc namespace.

Chapter 4

Common Attributes in OMDoc

Generally, the OMDoc format allows any attributes from foreign (i.e. non-OMDoc) namespaces on the OMDoc elements. This is a commonly found feature that makes the XML encoding of the OMDoc format extensible. Note that the attributes defined in this specification are in the default (empty) namespace: they do not carry a namespace prefix. So any attribute of the form `na:xxx` is allowed as long as it is in the scope of a suitable namespace prefix declaration.

Many OMDoc elements have optional `xml:id` attributes that can be used as identifiers to reference them. These attributes are of type `ID`, they must be unique in the document which is important, since many XML applications offer functionality for referencing and retrieving elements by `ID`-type attributes. Note that unlike other `ID`-attributes, in this special case it is the name `xml:id` [**XML:id05**] that defines the referencing and uniqueness functionality, not the type declaration in the DTD or XML schema (see for a discussion).

Note that in the OMDoc format proper, all ID type attributes are of the form `xml:id`. However in the older OpenMath and MathML standards, they still have the form `id`. The latter are only recognized to be of type `ID`, if a document type or XMLschema is present. Therefore it depends on the application context, whether a DTD should be supplied with the OMDoc document.

For many occasions (e.g. for printing OMDoc documents), authors want to control a wide variety of aspects of the presentation. OMDoc is a content-oriented format, and as such only supplies an infrastructure to mark up content-relevant information in OMDoc elements. To address this dilemma XML offers an interface to Cascading Style Sheets (CSS) [**BosHak:css98**], which allow to specify presentational traits like text color, font variant, positioning, padding, or frames of layout boxes, and even aural aspects of the text.

To make use of CSS, most OMDoc elements (all that have `xml:id` attributes) have `style` attributes that can be used to specify CSS directives for them. In the OMDoc fragment in Listing 4.1 we have used the `style` attribute to specify that the text content of the `omtext` element should be formatted in a centered box whose width is 80% of the surrounding box (probably the page box), and that has a 2 pixel wide solid frame of the specified RGB color. Generally CSS directives are of the form `A:V`, where `A` is the name of the aspect, and `V` is the value, several CSS directives can be combined in one `style` attribute as a semicolon-separated list (see [**BosHak:css98**] and the emerging CSS 3 standard).

Listing 4.1: Basic CSS Directives in a `style` Attribute

```
<?xml version="1.0" encoding="utf−8"?>
<?xml−stylesheet type="text/css" href="http://example.org/style.css"?>
<omdoc xml:id="stylish">
  ...
  <omtext xml:id="t1" style="width:80%;align:center;border:2px #006699 solid">
    <h:p>Here comes something
      <h:span style="font−weight:bold;color:green" class="emphasize">stylish</h:span>!
    </h:p>
  </omtext>
  ...
</omdoc>
```

Note that many CSS properties of parent elements are inherited by the children, if they are not explicitly specified in the child. We could for instance have set the font family of all the children of the `omtext` element by adding a directive `font-family:sans-serif` there and then override it by a directive for the property `font-family` in one of the children.

Frequently recurring groups of CSS directives can be given symbolic names in CSS styles heets, which can be referenced by the `class` attribute. In Listing 4.1 we have made use of this with the class `emphasize`, which we assume to be defined in the style sheet `style.css` associated with the document in the “style sheet processing instruction” in the prolog[1] of the XML document (see [**Clark:assxd99**] for details). Note that an OMDoc element can have both `class` and `style` attributes, in this case, precedence is determined by the rules for CSS style sheets as specified in [**BosHak:css98**]. In our example in Listing 4.1 the directives in the `style` attribute take precedence over the CSS directives in the style sheet referenced by the `class` attribute on the `phrase` element. As a consequence, the word “stylish” would appear in green, bold italics.

[1]i.e. at the very beginning of the XML document before the document type declaration