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

December 16, 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

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.

1 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.

1.1 Dimensions of Representation in OMDoc

BNP:1

Strict vs. Pragmatic The OMDoc format is divided into two sublanguages: "Strict" OMDoc (in the lower half of Figure 1) and "Pragmatic" OMDoc (in the upper half[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] In particular, the simplicity and conceptual clarity of strict OMDoc allow to express structural well-formedness constraints, whereas the vocabulary of pragmatic OMDoc is much nearer to mathematical practice and is thus easier to learn. It is a crucial design choice of the OMDoc format 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.

EdN:2

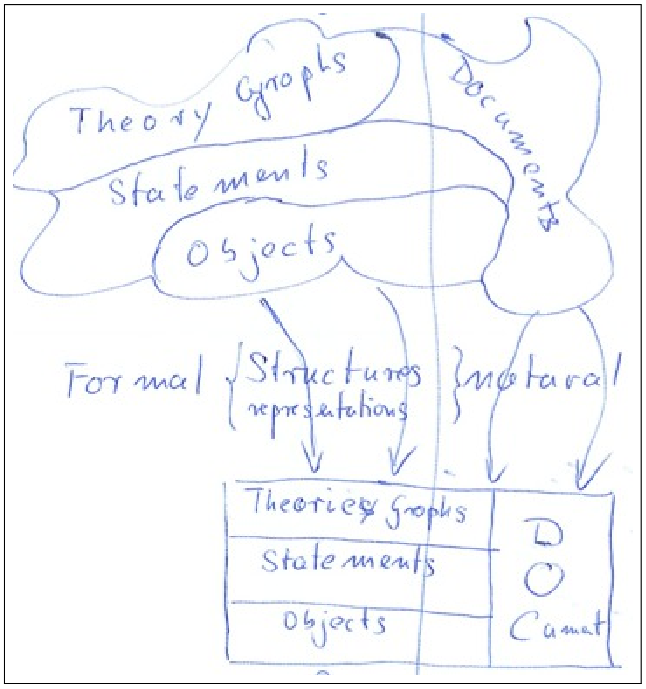

Figure 1: Dimensions of Representation in OMDoc

EdN:3

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

[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 [Aus+10]

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 representation formats for mathematical documents must allow this as well, consequently, OMDoc has two sub-languages, "formal OMDoc" (on the left side of Figure 1) and "natural OMDoc" (on the right side).

OMDoc offers markup at three levels: objects, statements, and context.

objects are usually represented as *formulae* or *natural language phrases* in mathematical documents. In formal OMDoc formulae 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 both ways and both ways of representation can be mixed at any level to account for mathematical practice, e.g. for mixed formulae like $\{n \in \mathbb{N} | n > 3 \text{ is prime}\}$.

statements are usually represented as *natural language sentences (with formulae)*[2] in informal settings and as (closed, logical) formulae in formal ones. The discussion about the two ways of representation of objects applies analogously. Note that functional markup in formal OMDoc only 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, OMDoc groups 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 it 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.

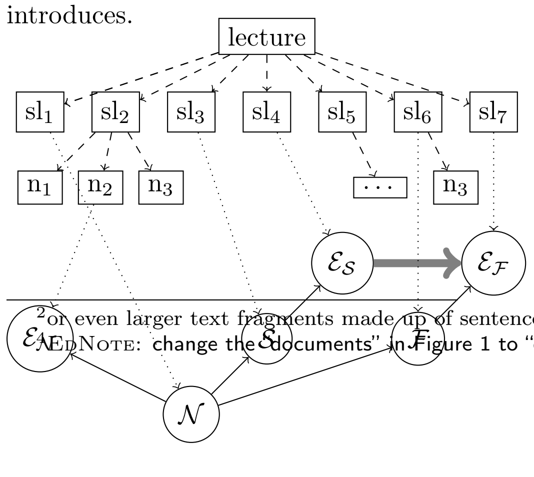

Figure 2: Content vs. Narrative Structures

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 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

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

[4] EdNote: change the "documents" in Figure 1 to "discourse", at least in the strict box

version $\mathcal{F}$ of the concept or theory and demonstrate that this works on $\mathcal{E}_{\mathcal{F}}$.

The structure with the solid lines and boxes at the bottom of Figure 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 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 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 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".

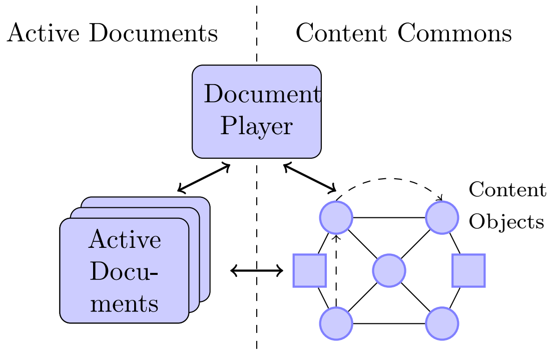

Figure 3: The Active Documents Paradigm

OMDoc aims 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 OMDoc is 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.

OMDoc extends 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 3 shows the [6]. EdN:6

Just as for content-based systems on the formula level, there are now MKM systems that generate presentation markup from content markup, based on general presentation principles, also on this level. For instance, the ActiveMath system [Mel+03] generates a simple narrative structure (the presentation; called a personalized book) from the underlying content structure (given in OMDoc) and a user model.

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

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

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

1.2 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 4 before we go into the specifics of the respective modules in Section 1 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 4 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?).

1.3 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[3], 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 [OMDoc].

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[4]:

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`	Subsubsection 2.1.1
OpenMath	`http://www.openmath.org/OpenMath`	Subsubsection 2.1.0
XSLT	`http://www.w3.org/1999/XSL/Transform`	?spec@pres?

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

[3]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.

[4]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.

Module	Title	Required?	Chapter
MOBJ	Mathematical Objects	yes	Section 1
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 4: The OMDoc Modules

```
<?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>
```

1.4 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` [MVW05] 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) [Bos+98], 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 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 [Bos+98] and the emerging CSS 3 standard).

Listing 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 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[5] of the XML document (see [Cla99a] 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 [Bos+98]. In our example in Listing 1 the directives in the `style` attribute take precedence over

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

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.

2 Mathematical Objects (Module MOBJ)

A distinguishing feature of mathematics is its ability to represent and manipulate ideas and objects in symbolic form as mathematical formulae. OMDoc uses the OpenMath and MathML formats to represent mathematical formulae and objects. Therefore, the OpenMath standard [Bus+04] and the MathML 3 recommendation [Aus+03] are part of this specification. OpenMath and MathML 3 are well-aligned: the Content-MathMLsublanguage directly encodes OpenMath objects. MathML additionally has a sublanguage for expressing the layout of formulae: Presentation-MathML, which can be mixed with Content-MathML; therefore we prefer MathML syntax for OMDoc and will use it throughout this specification. But we stress that OpenMath syntax is supported in OMDoc as well.

We will review OpenMath objects in Subsubsection 2.1.0 and Content-MathML in Subsubsection 2.1.1, and specify an OMDoc element for entering mathematical formulae (element `legacy`) in Subsection 2.6. [7] EdN:7

2.1 Content Representation of Mathematical Objects

We will now recapitulate the representational core of OpenMath and Content-MathML. Both represent mathematical objects as expressions made up from

1. **symbols** which reference previously declared mathematical objects, these are identified by referencing a content dictionary (see Definition 2.1).
2. **applications** of function or relation operators to a sequence of arguments
3. **binding structures** that represent functional objects with the help of **bound variables**.
4. **identifiers** for objects that are known locally (usually bound variables).

In Figure 7 we have put the OpenMath and strict Content-MathMLencodings of the law of commutativity for the real numbers side by side to show the similarities and differences. There is an obvious line-by-line similarity for the tree constructors and token elements. The main difference is the treatment of annotation, which we will describe in ‘Subsection 2.1.

Definition 2.1 A **content dictionary** (**CD**) is a machine-readable document that defines the meaning of mathematical concepts expressed by OpenMath/MathML symbols.

The OpenMath 2 standard provides a minimal data model and XML encoding for content dictionaries. We will not review the OpenMath content dictionary format there, since OMDoc theories fill that role in the OMDoc universe – see ?spec@identifying? for details.

The central idea is that symbols are identified by a triple:

1. the URI of the file containing the CD (called the **content dictionary base**),
2. the the name of the CD, called the **content dictionary name**, and
3. the local name in the CD, called the **symbol name**.

2.1.1 The Representational Core of OpenMath

OpenMath is a markup language for mathematical formulae that concentrates on the meaning of formulae building on an extremely simple kernel (markup primitive for syntactical forms of content formulae), and adds an extension mechanism for mathematical concepts, the content dictionaries.

We will only review the XML encoding of OpenMath objects here, since it is most relevant to the OMDoc format. All elements of the XML encoding live in the namespace `http://www.openmath.org/OpenMath`, for which we traditionally use the namespace prefix `om:`.

Definition 2.2 The **om:OMA** element contains representations of the function and its argument in “prefix-” or “Polish notation”, i.e. the first child is the representation of the function and all the subsequent ones are representations of the arguments in order. `om:OMA`

Element	Attributes		Content
	Required	Optional	
om:OMS	cd, name	id, cdbase, class, style	EMPTY
om:OMV	name	id, class, style	EMPTY
om:OMA		id, cdbase, class, style	⟪*OMel*⟫*
om:OMBIND		id, cdbase, class, style	⟪*OMel*⟫,OMBVAR,⟪*OMel*⟫
om:OMBVAR		id, class, style	(OMV | OMATTR)+
om:OMFOREIGN		id, cdbase, class, style	ANY
om:OMATTR		id, cdbase, class, style	⟪*OMel*⟫
om:OMATP		id, cdbase, class, style	(OMS, (⟪*OMel*⟫|OMFOREIGN))+
om:OMI		id, class, style	[0-9]*
om:OMB		id, class, style	#PCDATA
om:OMF		id, class, style, dec, hex	#PCDATA
om:OME		id, class, style	⟪*OMel*⟫?
om:OMR	href		⟪*OMel*⟫?
where ⟪*OMel*⟫ is (OMS|OMV|OMI|OMB|OMSTR|OMF|OMA|OMBIND|OME|OMATTR)			

Figure 5: OpenMath Objects in OMDoc

Definition 2.3 Objects and concepts that carry meaning independent of the local context (they are called **symbols**) in OpenMath) are represented as **om:OMS** elements, where the value of the `name` attribute gives the name of the symbol. The `cd` attribute specifies the relevant content dictionary, the optional `cdbase` on an `om:OMS` element contains a URI that can be used to disambiguate the content dictionary. Alternatively, the `cdbase` attribute can be given on an OpenMath element that is a parent to the `om:OMS` in question: The `om:OMS` inherits the `cdbase` of the nearest ancestor (inducing the usual XML scoping rules for declarations).[6]

om:OMS

Definition 2.4 Variables are represented as **om:OMV** element. As variables do not carry a meaning independent of their local content, `om:OMV` only carries a `name` attribute (see Subsection 2.5 for further discussion).

om:OMV

Example 2.5 The formula sin(x) would be modeled as an application of the sin function (which in turn is represented as an OpenMath symbol) to a variable:

```
<OMA xmlns="http://www.openmath.org/OpenMath"
     cdbase="http://www.openmath.org/cd">
  <OMS cd="transc1" name="sin"/>
  <OMV name="x"/>
</OMA>
```

In our case, the function sin is represented as an `om:OMS` element with name `sin` from the content dictionary `transc1`. The `om:OMS` inherits the `cdbase`-value `http://www.openmath.org/cd`, which shows that it comes from the OpenMath standard collection of content dictionaries from the `om:OMA` element above. The variable x is represented in an `om:OMV` element with `name`-value `x`.

Example 2.6 For the `om:OMBIND` element consider the following representation of the formula $\forall x.\sin(x) \leq \pi$.

```
<OMBIND xmlns="http://www.openmath.org/cd">
  <OMS cd="quant1" name="forall"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMA>
    <OMS cd="arith1" name="leq"/>
    <OMA><OMS cd="transc1" name="sin"/><OMV name="x"/></OMA>
    <OMS cd="nums1" name="pi"/>
  </OMA>
</OMBIND>
```

Definition 2.7 The **om:OMBIND** element has exactly three children, the first one is a "binding

om:OMBIND

[7]EdNote: discuss MathML3 and the relation between MathML and OpenMath and what that means for OMDoc

[6]Note that while the `cdbase` inheritance mechanism described here remains in effect for OpenMath objects embedded in to the OMDoc format, it is augmented by one in OMDoc. As a consequence, OpenMath objects in OMDoc documents will usually not contain `cdbase` attributes; see ?spec@identifying? for a discussion.

operator"[7] — in this case the universal quantifier, the second one is a list of bound variables that must be encapsulated in an **om:OMBVAR** element, and the third is the body of the binding object, in which the bound variables can be used. OpenMath uses the `om:OMBIND` element to unambiguously specify the scope of bound variables in expressions: the bound variables in the `om:OMBVAR` element can be used only inside the mother `om:OMBIND` element, moreover they can be systematically renamed without changing the meaning of the binding expression. As a consequence, bound variables in the scope of an `om:OMBIND` are distinct as OpenMath objects from any variables outside it, even if they share a name.

`om:OMBVAR`

2.1.2 Strict Content MathML

Content-MathMLis a content markup format that represents the abstract structure of formulae in trees of logical sub-expressions much like OpenMath: the MathML 3 recommendation [Aus+10] identifies a sublanguage: strict Content-MathMLthat is isomorphic to OpenMath 2.

Element	Attributes		Content
	Required	Optional	
`m:math`		`id`, `xlink:href`	⟪*CMel*⟫+
`m:apply`		`id`, `xlink:href`	`m:bvar?`,⟪*CMel*⟫*
`m:csymbol`	`definitionURL`	`id`, `xlink:href`	EMPTY
`m:ci`		`id`, `xlink:href`	#PCDATA
`m:cn`		`id`, `xlink:href`	([0-9]|,|.)(*|e([0-9]|,|.)*)?
`m:bvar`		`id`, `xlink:href`	`m:ci`| `m:semantics`
`m:semantics`		`id`, `xlink:href`, `definitionURL`	⟪*CMel*⟫,(`m:annotation` | `m:annotation-xml`)*
`m:annotation`		`definitionURL`, `encoding`	#PCDATA
`m:annotation-xml`		`definitionURL`, `encoding`	ANY
where ⟪*CMel*⟫ is `m:apply`| `m:csymbol`| `m:ci`| `m:cn`|`m:semantics`			

Figure 6: Content-MathML in OMDoc

Definition 2.8 The top-level element of MathML is the **m:math** element it contains an functional expression composed

1. identifiers (element **m:ci**) corresponding to variables. The content of the `m:ci` element is a unicode string used as the name of the identifier.
2. symbols (element **m:csymbol**) for arbitrary symbols. The content of the `m:csymbol` element is the name of the symbol, its meaning is dermined by the `csymbol` attribute that contains a content dictionary name.
3. applications (element**m:apply**)
4. binding structures (element **m:bind** with **m:bvars**).

`m:math` `m:ci` `m:csymbol` `m:apply` `m:bind` `m:bvars`

2.2 Annotating Mathematical Objects

OpenMath offers an element for annotating (parts of) formulae with external information (e.g. MathML or LaTeX presentation):

Definition 2.9 The **om:OMATTR** element that pairs an OpenMath object with an attribute-value list. To annotate an OpenMath object, it is embedded as the second child in an `om:OMATTR` element. The attribute-value list is specified by children of the preceding **om:OMATP** (Attribute value Pair) element, which has an even number of children: children at odd positions must be

`om:OMATTR` `om:OMATP`

[7]The binding operator must be a symbol which either has the role `binder` assigned by the OpenMath content dictionary (see [Bus+04] for details) or the symbol declaration in the OMDoc content dictionary must have the value `binder` for the attribute `role` (see ?spec@symbol-dec?).

OpenMath

```
<OMBIND>
 <OMS cd="quant1" name="forall"/>
 <OMBVAR>
  <OMATTR>
   <OMATP>
    <OMS cd="sts" name="type"/>
    <OMS cd="setname1" name="R"/>
   </OMATP>
   <OMV name="a"/>
  </OMATTR>


  <OMATTR>
   <OMATP>
    <OMS cd="sts" name="type"/>
    <OMS cd="setname1" name="R"/>
   </OMATP>
   <OMV name="b"/>
  </OMATTR>
 </OMBVAR>
  <OMA>
   <OMS cd="relation" name="eq"/>
   <OMA>
    <OMS cd="arith1" name="plus"/>
    <OMV name="a"/>
    <OMV name="b"/>
   </OMA>
   <OMA>
    <OMS cd="arith1" name="plus"/>
    <OMV name="b"/>
    <OMV name="a"/>
   </OMA>
 </OMA>
</OMBIND>
```

MathML

```
<m:apply>
 <m:forall/>
 <m:bvar>
  <m:semantics>
   <m:ci>a</m:ci>
   <m:annotation−xml
     definitionURL="http://www.openmath.org/cd/sts#type">
    <m:csymbol cd="setname1">R</m:csymbol>
   </m:annotation−xml>
  </m:semantics>
 </m:bvar>
 <m:bvar>
  <m:semantics>
   <m:ci>a</m:ci>
   <m:annotation−xml
     definitionURL="http://www.openmath.org/cd/sts#type">
    <m:csymbol cd="setname1">R</m:csymbol>
   </m:annotation−xml>
  </m:semantics>
 </m:bvar>
 <m:apply>
  <m:csymobl cd="relation1">eq</m:csymbol>
  <m:apply>
   <m:csymbol cd="arith1">plus</m:csymbol>
   <m:ci>a</m:ci>
   <m:ci>b</m:ci>
  </m:apply>
  <m:apply>
   <m:csymbol cd="arith1">plus</m:csymbol>
   <m:ci>b</m:ci>
   <m:ci>a</m:ci>
  </m:apply>
 </m:apply>
</m:apply>
```

Figure 7: OpenMath vs. C-MathML for Commutativity

`om:OMS` (specifying the attribute, they are called **keys** or **features**)[8], and children at even positions are the **values** of the keys specified by their immediately preceding siblings. In the OpenMath fragment in Listing 2 the expression $x + \pi$ is annotated with an alternative representation and a color.

A special application of the `om:OMATTR` element is associating non-OpenMath objects with OpenMath objects.

Definition 2.10 For this, OpenMath2 allows to use an **om:OMFOREIGN** element in the even positions of an `om:OMATP`. This element can be used to hold arbitrary XML content (in our example above SVG: Scalable Vector Graphics [JFF02]), its required `encoding` attribute specifies the format of the content. `om:OMFOREIGN`

We recommend a MIME type [FB96] (see ?spec@pres-bound? for an application).

Example 2.11 Here we use the `om:OMATTR` element to associate various other representationsn with an application.

Listing 2: Associating Alternate Representations with an OpenMath Object

```
<OMATTR>
  <OMATP>
    <OMS cd="alt−rep" name="ascii"/>
```

[8]There are two kinds of keys in OpenMath distinguished according to the `role` value on their `symbol` declaration in the content dictionary: `attribution` specifies that this attribute value pair may be ignored by an application, so it should be used for information which does not change the meaning of the attributed OpenMath object. The `role` is used for keys that modify the meaning of the attributed OpenMath object and thus cannot be ignored by an application.

```
    <OMSTR>(x+1)</OMSTR>
    <OMS cd="alt−rep" name="svg"/>
    <OMFOREIGN encoding="application/svg+xml">
      <svg xmlns='http://www.w3.org/2000/svg'>. . .</svg>
    </OMFOREIGN>
    <OMS cd="pres" name="color"/>
    <OMS cd="pres" name="red"/>
  </OMATP>
  <OMA>
    <OMS cd="arith1" name="plus"/>
    <OMV name="x"/>
    <OMS cd="nums1" name="pi"/>
  </OMA>
</OMATTR>
```

In Content-MathML, the same effect can be achieved by the `m:semantics` element whose first child is the annotated object and subsequent `m:annotation` and `m:annotation-xml` and children carry the annotations.

Definition 2.12 The **m:semantics** element provides a way to annotate Content-MathML elements with arbitrary information. The first child of the `m:semantics` element is annotated with the information in the **m:annotation-xml** (for XML-based information) and **m:annotation** (for other information) elements that follow it. These elements carry `definitionURL` attributes that point to a "definition" of the kind of information provided by them. The optional `encoding` is a string that describes the format of the content.

`m:semantics`

`m:annotation-xml`

`m:annotation`

Example 2.13 To express the content of Example 2.11 in Content-MathML, we use the `m:semantics`, `m:annotation`, and `m:annotationxml` elements.

Listing 3: Associating Alternate Representations in Content-MathML

```
<m:semantics>
  <m:apply>
    <m:csymbol cd="arith1">plus</m:csymbol>
    <m:ci>x</m:ci>
    <m:csymbol cd="nums1">pi</m:csymbol>
  </m:apply>
  <m:annotation definitionURL="http://omdoc.org/cds/alt−rep#ascii">
    (x+1)
  </m:annotation>
  <m:annotation−xml definitionURL="http://omdoc.org/cds/alt−rep#svg"
      encoding="application/svg+xml">
   <svg xmlns='http://www.w3.org/2000/svg'>. . .</svg>
  </m:annotation−xml>
  <m:annotation−xml definitionURL="http://omdoc.org/cds/pres#color"
      encoding="application/openmath+xml">
   <OMS cd="pres" name="red"/>
  </m:annotation−xml>
</m:semantics>
```

2.3 Parallel Markup

The idea of parallel markup has been pioneered in the MathML language [Aus+10]. For parallel markup of a formula, MathML combines the presentation and content trees in a single XML tree and marks up corresponding subtrees by cross-references. Parallel markup supports two workflows:

1. *formalization*: the annotation of presentation formulae with (multiple) formalizations and
2. *presentation*: the annotation of content formulae with (multiple) presentations.

Let us look at an well-known example: The (presentation) formula $f(a+b)$ can be interpreted in two ways: as the application of the function f to the sum $a+b$ or as the product of a scalar f with the sum $a+b$. Consequently, we can annotate the presentation tree of $f(a+b)$ with two content trees to arrive at the situation depicted in Figure 8. In MathML, parallel markup is implemented by the `semantics` element, which has the central element as the first child and the annotations in subsequent `annotation−xml` children. We will disregard this technical level here and concentrate on the concepts here.

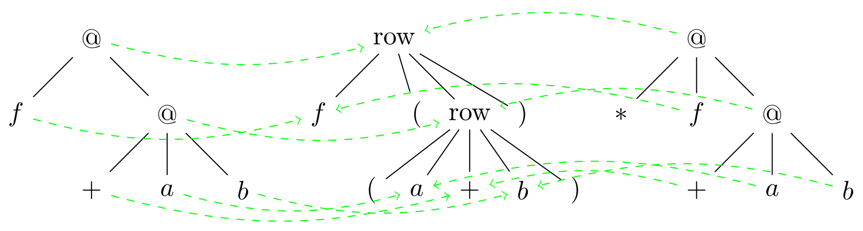

Figure 8: A presentation tree (center) with content annotations

Conversely for the presentation workflow, the product of f with $a + b$ can be presented as $f(a + b)$ (as above) but also with $f \cdot (a + b)$, so that we arrive in the situation in Figure 9. In both situations, there is a central tree annotated with further representations in the respective other language. Which of the situations is used depends on the context. In the content commons we would expect to see content trees annotated with presentations (as in Figure 9) and in the document commons the other way around (as in Figure 8). Note also that the direction of the arrows is always from the (possibly multiple) annotations into the center.

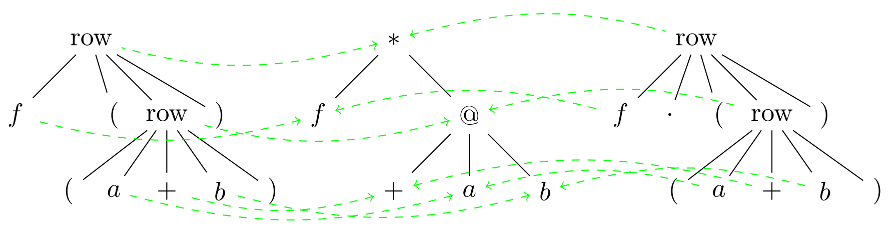

Figure 9: A content tree (center) with presentation annotations

2.4 Mixing Formal and Informal Markup in Objects

[8] EdN:8

Example 2.14 The formula $\{x \in \mathbb{N} \mid |x$ is prime and $x > 2\}$ can be represented as

```
<m:bind>
 <m:apply>
  <m:csymbol cd="sets">setst</m:csymbol>
  <m:csymbol cd="numbersets">N</m:csymbol>
 </m:apply>
 <m:bvars><m:ci>x<m:ci></m:bvars>
 <m:semantics>
  <m:apply id="a">
   <m:ci id="ip">ipa</m:ci>
   <m:ci id="x1">x<m:ci>
   <m:apply id="xgt2">
    <m:csymbol cd="relation1" id="gt">greater</m:csymbol>
    <m:ci id="x2">x<m:ci>
    <m:cn id="n3">2</m:cn>
   </m:apply>
  </m:apply>
  <m:annotation−xml encoding="pMathML">
   <m:mrow xref="#a">
    <m:mi xref="#x1">x</m:mi>
    <m:mtext xref="#ip">is prime and</m:mtext>
    <m:mrow xref="#xgt3">
     <m:mi xref="#x2">x</m:mi>
     <m:mo xref="gt">&gt;</m:mo>
     <m:mn xref="n2">2</m:mn>
    </m:mrow>
   </m:mrow>
```

[8] EdNote: Intro here; reference spec-intro

```
  </m:annotation-xml>
 <m:semantics>
</m:bind>
```

We use content markup for the set construct and then use parallel markup for its body: x is prime. The content part is partially opaque to Content-MathML[9], so we represent its as an application of an unspecified predicate ipa applied to the content-regognizable parts x and $x > 2$. On the presentation side, we have the obvious markup, which cross-references into the content for parallel markup. Note that we are not assuming any fancy natural language processing here, which might have spotted the fact that we have a conjunction of two atomic requirements "x is prime" and "$x > 2$, which would have led to a more strutured Content-MathMLexpression. Instead the cross-references are established in a bottom-up manner; referencing corresponding sub-expressions and defaulting to the unspecified predicate ipa in doubt. EdN:9

2.5 Programming Extensions of for Mathematical Objects

Definition 2.15 For representing objects in computer algebra systems OpenMath also provides other basic data types: **om:OMI** for integers, **om:OMB** for byte arrays, **om:OMSTR** for strings, and **om:OMF** for floating point numbers. These do not play a large role in the context of OMDoc, so we refer the reader to the OpenMath standard [Bus+04] for details.

om:OMI

om:OMB

om:OMSTR

om:OMF

Definition 2.16 Content-MathMLuses the **m:cn** element for number expressions. The attribute `type` can be used to specify the mathematical type of the number, e.g. `complex`, `real`, or `integer`. The content of the `m:cn` element is interpreted as the value of the number expression.

m:cn

Definition 2.17 The **om:OME** element is used for in-place error markup in OpenMath objects, it can be used almost everywhere in OpenMath elements. It has two children; the first one is an error operator[9], i.e. an OpenMath symbol that specifies the kind of error, and the second one is the faulty OpenMath object fragment. Note that since the whole object must be a valid OpenMath object, the second child must be a well-formed OpenMath object fragment.

om:OME

As a consequence, the `om:OME` element can only be used for "semantic errors" like non-existing content dictionaries, out-of-bounds errors, etc. XML-well-formedness and DTD-validity errors will have to be handled by the XML tools involved. In the following example, we have marked up two errors in a faulty representation of $\sin(\pi)$. The outer error flags an arity violation (the function sin only allows one argument), and the inner one flags the typo in the representation of the constant π (we used the name `po` instead of `pi`).

```
<OME>
  <OMS cd="type-error" name="arity-violation"/>
  <OMA>
    <OMS cd="transc1" name="sin"/>
    <OME>
      <OMS cd="error" name="unexpected_symbol"/>
      <OMS cd="nums1" name="po"/>
    </OME>
    <OMV name="x"/>
  </OMA>
</OME>
```

As we can see in this example, errors can be nested to encode multiple faults found by an OpenMath application.

[10] EdN:10

[9]EdNote: explain opaque and/or intro above

[9]An error operator is like a binding operator, only the symbol has role `error`.

[10]EdNote: need to talk about the `m:cerror` element

2.6 The Semantics of Variables in OpenMath and Content-MathML

A more subtle, but nonetheless crucial difference between OpenMath and MathML is the handling of variables, symbols, their names, and equality conditions. OpenMath uses the `name` attribute to identify a variable or symbol, and delegates the presentation of its name to other methods such as style sheets. As a consequence, the elements `om:OMS` and `om:OMV` are empty, and we have to understand the value of the `name` attribute as a pointer to a defining occurrence. In case of symbols, this is the symbol declaration in the content dictionary identified in the `cd` attribute. A symbol `<OMS cd="`$\langle\!\langle cd_1\rangle\!\rangle$`" name="`$\langle\!\langle name_1\rangle\!\rangle$`"/>` is equal to `<OMS cd="`$\langle\!\langle cd_2\rangle\!\rangle$`" name="`$\langle\!\langle name_2\rangle\!\rangle$`"/>`, iff $\langle\!\langle cd_1\rangle\!\rangle=\langle\!\langle cd_2\rangle\!\rangle$ and $\langle\!\langle name_1\rangle\!\rangle=\langle\!\langle name_2\rangle\!\rangle$ as XML simple names. In case of variables this is more difficult: if the variable is bound by an `om:OMBIND` element We say that an `om:OMBIND` element **binds** an OpenMath variable `<OMV name="x"/>`, iff this `om:OMBIND` element is the nearest one, such that `<OMV name="x"/>` occurs in (second child of the `om:OMATTR` element in) the `om:OMBVAR` child (this is the **defining occurrence** of `<OMV name="x"/>` here)., then we interpret all the variables `<OMV name="x"/>` in the `om:OMBIND` element as equal and different from any variables `<OMV name="x"/>` outside. In fact the OpenMath standard states that bound variables can be renamedo without changing the object (α-conversion). If `<OMV name="x"/>` is not bound, then the scope of the variable cannot be reliably defined; so equality with other occurrences of the variable `<OMV name="x"/>` becomes an ill-defined problem. We therefore discourage the use of unbound variables in OMDoc; they are very simple to avoid by using symbols instead, introducing suitable theories if necessary (see ?spec@theories-contexts?).

MathML goes a different route: the `m:csymbol` and `m:ci` elements have content that is Presentation-MathML, which is used for the presentation of the variable or symbol name.[10] While this gives us a much better handle on presentation of objects with variables than OpenMath (where we are basically forced to make due with the ASCII[11] representation of the variable name), the question of scope and equality becomes much more difficult: Are two variables (semantically) the same, even if they have different colors, sizes, or font families? Again, for symbols the situation is simpler, since the `definitionURL` attribute on the `m:csymbol` element establishes a global identity criterion (two symbols are equal, iff they have the same `definitionURL` value (as URI strings; see [BFM98]).) The second edition of the MathML standard adopts the same solution for bound variables: it recommends to annotate the `m:bvar` elements that declare the bound variable with an `id` attribute and use the `definitionURL` attribute on the bound occurrences of the `m:ci` element to point to those. The following example is taken from [KD03], which has more details.

```
<m:lambda>
  <m:bvar><m:ci xml:id="the−boundvar">complex presentation</m:ci></m:bvar>
  <m:apply>
    <m:plus/>
    <m:ci definitionURL="#the−boundvar">complex presentation</m:ci>
    <m:ci definitionURL="#the−boundvar">complex presentation</m:ci>
  </m:apply>
</m:lambda>
```

For presentation in MathML, this gives us the best of both approaches, the `m:ci` content can be used, and the pointer gives a simple semantic equivalence criterion. For presenting OpenMath and Content-MathML in other formats OMDoc makes use of the infrastructure introduced in module PRES; see ?spec@pres-bound? for a discussion.

2.7 Legacy Representation for Migration

Sometimes, OMDoc is used as a migration format from legacy texts (see [Koh09a, Part I] for an example). In such documents it can be too much effort to convert all mathematical objects and formulae into OpenMath or Content-MathML form.

[10] Note that surprisingly, the empty Content-MathML elements are treated more in the OpenMath spirit.

[11] In the current OpenMath standard, variable names are restricted to alphanumeric characters starting with a letter. Note that unlike with symbols, we cannot associate presentation information with variables via style sheets, since these are not globally unique (see ?spec@pres-bound? for a discussion of the OMDoc solution to this problem).

Element	Attributes		Content
	Required	Optional	
`legacy`	`format`	`xml:id`, `formalism`	#PCDATA

Figure 10: Mathematical Objects in OMDoc

Definition 2.18 For this situation OMDoc provides the **legacy** element, which can contain arbitrary math markup[12]. The `legacy` element can occur wherever an OpenMath object or Content-MathML expression can and has an optional `xml:id` attribute for identification. The content is described by a pair of attributes: `legacy`

- `format` (required) specifies the format of the content using URI reference. OMDoc does not restrict the possible values, possible values include `TeX`, `pmml`, `html`, and `qmath`.
- `formalism` is optional and describes the formalism (if applicable) the content is expressed in. Again, the value is unrestricted character data to allow a URI reference to a definition of a formalism.

For instance in the following `legacy` element, the identity function is encoded in the untyped λ-calculus, which is characterized by a reference to the relevant Wikipedia article.

```
<legacy format="TeX" formalism="http://en.wikipedia.org/wiki/Lambda_calculus">
  \lambda{x}{x}
</legacy>
```

References

[ABD03] Andrea Asperti, Bruno Buchberger, and James Harold Davenport, eds. *Mathematical Knowledge Management, MKM'03.* LNCS 2594. Springer Verlag, 2003.

[Adi+08] Ben Adida et al. *RDFa in XHTML: Syntax and Processing.* W3C Recommendation. World Wide Web Consortium (W3C), Oct. 2008. URL: `http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/`.

[AK02] Andrea Asperti and Michael Kohlhase. "MathML in the MoWGLI Project". In: *Second International Conference on MathML and Technologies for Math on the Web.* Chicago, USA, 2002. URL: `http://www.mathmlconference.org/2002/presentations/asperti/`.

[AKS03] Andrea Asperti, Michael Kohlhase, and Claudio Sacerdoti Coen. *Prototype n. D2.b Document Type Descriptors: OMDoc Proofs.* MoWGLI Deliverable. The MoWGLI Project, 2003.

[Asp+01] Andrea Asperti et al. "HELM and the Semantic Math-Web". In: *Theorem Proving in Higher Order Logics: TPHOLs'01.* Ed. by Richard. J. Boulton and Paul B. Jackson. LNCS 2152. Springer Verlag, 2001, pp. 59–74.

[Aus+03] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 2.0 (second edition).* W3C Recommendation. World Wide Web Consortium (W3C), 2003. URL: `http://www.w3.org/TR/MathML2` (cit. on p. 10).

[Aus+10] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0.* Tech. rep. World Wide Web Consortium (W3C), 2010. URL: `http://www.w3.org/TR/MathML3` (cit. on pp. 3, 12, 14).

[Aut+00] Serge Autexier et al. "Towards an Evolutionary Formal Software-Development Using CASL". In: *Proceedings Workshop on Algebraic Development Techniques, WADT-99.* Ed. by C. Choppy and D. Bert. LNCS 1827. Springer Verlag, 2000, pp. 73–88.

[12] If the content is an XML-based, format like Scalable Vector Graphics [JFF02], the DTD must be augmented accordingly for validation.

[Bau99] Judith Baur. "Syntax und Semantik mathematischer Texte – ein Prototyp". MA thesis. SaarbrückenGermany: Fachrichtung Computerlinguistik, Universität des Saarlandes, 1999.

[BC01] Henk Barendregt and Arjeh M. Cohen. "Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants". In: *Journal of Symbolic Computation* 32 (2001), pp. 3–22.

[Ben+97] Christoph Benzmüller et al. "ΩMEGA: Towards a mathematical assistant". In: *Proceedings of the 14th Conference on Automated Deduction.* Ed. by William McCune. LNAI 1249. Townsville, Australia: Springer Verlag, 1997, pp. 252–255. URL: `http://kwarc.info/kohlhase/papers/Omega97-CADE.pdf`.

[Ber91] Paul Bernays. *Axiomatic Set Theory.* Dover Publications, 1991.

[BFM98] Tim Berners-Lee, Roy T. Fielding, and Larry. Masinter. *Uniform Resource Identifiers (URI), Generic Syntax.* RFC 2717. Internet Engineering Task Force (IETF), 1998. URL: `http://www.ietf.org/rfc/rfc2717.txt` (cit. on p. 17).

[BM04] Paul V. Biron and Ashok Malhotra. *XML Schema Part 2: Datatypes Second Edition.* Tech. rep. World Wide Web Consortium (W3C), Oct. 28, 2004. URL: `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`.

[Bos+98] *Cascading Style Sheets, level 2; CSS2 Specification.* W3C Recommendation. World Wide Web Consortium (W3C), 1998. URL: `http://www.w3.org/TR/1998/REC-CSS2-19980512` (cit. on p. 8).

[Bra+02] R. Bradford et al. "Reasoning About the Elementary Functions of Complex Analysis." In: *Annals of Mathematics and Artificial Intelligence* 36 (2002), pp. 303–318.

[Bra+04] Tim Bray et al. *Extensible Markup Language (XML) 1.1.* W3C Recommendation REC-xml11-20040204. World Wide Web Consortium (W3C), 2004. URL: `http://www.w3.org/TR/2004/REC-xml11-20040204/`.

[Bru94] Nicolaas Govert de Bruijn. "The Mathematical Vernacular, A Language for Mathematics with Typed Sets". In: *Selected Papers on Automath.* Ed. by R. P Nederpelt, J. H. Geuvers, and R. C. de Vrijer. Vol. 133. Studies in Logic and the Foundations of Mathematics. Elsevier, 1994, pp. 865–935.

[Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0.* Tech. rep. The OpenMath Society, 2004. URL: `http://www.openmath.org/standard/om20` (cit. on pp. 10, 12, 16).

[CCL] *About the Licenses – Creative Commons.* URL: `http://creativecommons.org/licenses` (visited on 05/31/2013).

[CCM] *Metadata Commons Worldwide.* URL: `http://creativecommons.org/learn/technology/metadata` (visited on 11/02/2014).

[CCW] *Creative Commons Worldwide.* URL: `http://creativecommons.org/worldwide` (visited on 11/02/2014).

[Cla+03] Edmund Clarke et al. "System Description: Analytica 2". In: *11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus 2003).* Ed. by Thérèse Hardin and Renaud Rioboo. Rome, Italy, Sept. 2003, pp. 69–74. URL: `http://kwarc.info/kohlhase/papers/calculemus03.pdf`.

[Cla99a] *Associating Style Sheets with XML Documents Version 1.0.* W3C Recommendation. World Wide Web Consortium (W3C), 1999. URL: `http://www.w3.org/TR/xml-stylesheet` (cit. on p. 8).

[Cla99b] *XSL Transformations (XSLT) Version 1.0.* W3C Recommendation. World Wide Web Consortium (W3C), 1999. URL: `http://www.w3.org/TR/xslt`.

[Con+86] Robert L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Englewood Cliffs, NJUSA: Prentice-Hall, 1986.

[DCM03a] The DCMI Usage Board. *DCMI Metadata Terms.* DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: `http://dublincore.org/documents/dcmi-terms/`.

[DCM03b] The DCMI Usage Board. *DCMI Type Vocabulary.* DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: `http://dublincore.org/documents/dcmi-type-vocabulary/`.

[DeR+01] Steve DeRose et al. *XML Linking Language (XLink Version 1.0).* W3C Recommendation. World Wide Web Consortium (W3C), 2001. URL: `http://www.w3.org/TR/2000/REC-xlink-20010627/`.

[DW05] Mark Davis and Ken Whistler. *Unicode Collation Algorithm.* Unicode Technical Standard #10. 2005. URL: `http://www.unicode.org/reports/tr10/`.

[Far93] William M. Farmer. "Theory Interpretation in Simple Type Theory". In: *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting.* LNCS 816. Amsterdam, The Netherlands: Springer Verlag, 1993.

[FB96] N. Freed and N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.* RFC 2046: `http://www.faqs.org/rfcs/rfc2046.html`. 1996. URL: `http://www.faqs.org/rfcs/rfc2046.html` (cit. on p. 13).

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. "IMPS: An Interactive Mathematical Proof System". In: *Journal of Automated Reasoning* 11.2 (Oct. 1993), pp. 213–248.

[Fie97] Armin Fiedler. "Towards a Proof Explainer". In: *Proceedings of the First International Workshop on Proof Transformation and Presentation.* Ed. by J. Siekmann, F. Pfenning, and X. Huang. Schloss DagstuhlGermany, 1997, pp. 53–54.

[Gen35] Gerhard Gentzen. "Untersuchungen über das logische Schließen I & II". In: *Mathematische Zeitschrift* 39 (1935), pp. 176–210, 572–595.

[Gog+03] George Goguadze et al. "Problems and Solutions for Markup for Mathematical Examples and Exercises". In: *Mathematical Knowledge Management, MKM'03.* Ed. by Andrea Asperti, Bruno Buchberger, and James Harold Davenport. LNCS 2594. Springer Verlag, 2003, pp. 80–93.

[Gro+07] Tudor Groza et al. "SALT – Semantically Annotated LaTeX for Scientific Publications". In: *The Semantic Web: Research and Applications.* 4th European Semantic Web Conference (ESWC). (Innsbruck, Austria, June 3–7, 2007). Ed. by Enrico Franconi, Michael Kifer, and Wolfgang May. LNCS 4519. Springer Verlag, 2007, pp. 518–532. ISBN: 9783540726661.

[Gro99] The Open eBook Group. *Open eBook[tm] Publication Structure 1.0.* Draft Recommendation. The OpenEBook Initiative, 1999. URL: `http://www.openEbook.org`.

[Har03] Eliotte Rusty Harold. "Effective XML". In: Addison Wesley, 2003. Chap. 15.

[HF96] Xiaorong Huang and Armin Fiedler. "Presenting Machine-Found Proofs". In: *Proceedings of the 13th Conference on Automated Deduction.* Ed. by M. A. McRobbie and J. K. Slaney. LNAI 1104. New Brunswick, NJ, USA: Springer Verlag, 1996, pp. 221–225.

[Hut00] Dieter Hutter. "Management of Change in Verification Systems". In: *Proceedings 15th IEEE International Conference on Automated Software Engineering, ASE-2000.* IEEE Computer Society, 2000, pp. 23–34.

[Inc03] Unicode Inc., ed. *The Unicode Standard, Version 4.0.* Addison-Wesley, 2003.

[JFF02] Dean Jackson, Jon Ferraiolo, and Jun Fujisawa. *Scalable Vector Graphics (SVG) 1.1 Specification.* W3C Candidate Recommendation. World Wide Web Consortium (W3C), Apr. 2002. URL: `http://www.w3.org/TR/2002/CR-SVG11-20020430` (cit. on pp. 13, 18).

[KA03] Michael Kohlhase and Romeo Anghelache. “Towards Collaborative Content Management And Version Control For Structured Mathematical Knowledge”. In: *Mathematical Knowledge Management, MKM’03*. Ed. by Andrea Asperti, Bruno Buchberger, and James Harold Davenport. LNCS 2594. Springer Verlag, 2003, pp. 147–161. URL: `http://kwarc.info/kohlhase/papers/mkm03.pdf`.

[KD03] Michael Kohlhase and Stan Devitt. *Bound Variables in MathML*. W3C Working Group Note. 2003. URL: `http://www.w3.org/TR/mathml-bvar/` (cit. on p. 17).

[KK06] Andrea Kohlhase and Michael Kohlhase. “An Exploration in the Space of Mathematical Knowledge”. In: *Mathematical Knowledge Management, MKM’05*. Ed. by Michael Kohlhase. LNAI 3863. Springer Verlag, 2006, pp. 17–32. URL: `http://kwarc.info/kohlhase/papers/mkm05.pdf`.

[KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. “Notations for Living Mathematical Documents”. In: *Intelligent Computer Mathematics*. 9th International Conference, AISC, 15th Symposium, Calculemus, 7th International Conference MKM. (Birmingham, UK, July 28–Aug. 1, 2008). Ed. by Serge Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 504–519. URL: `http://omdoc.org/pubs/mkm08-notations.pdf`.

[Koh] Michael Kohlhase. “CodeML: An Open Markup Format the Content and Presentatation of Program Code”. Internet Draft at `https://svn.omdoc.org/repos/codeml/doc/spec/codeml.pdf`. URL: `https://svn.omdoc.org/repos/codeml/doc/spec/codeml.pdf`.

[Koh05] Michael Kohlhase. *Inference Rules*. OMDoc Content Dictionary at `https://svn.omdoc.org/repos/omdoc/trunk/examples/logics/inference-rules.omdoc`. seen Jan 2005. URL: `https://svn.omdoc.org/repos/omdoc/trunk/examples/logics/inference-rules.omdoc`.

[Koh06] Michael Kohlhase, ed. *Mathematical Knowledge Management, MKM’05*. LNAI 3863. Springer Verlag, 2006.

[Koh09a] Michael Kohlhase. “An OMDoc Primer [Version 1.6 (pre-2.0)]”. Draft `https://svn.omdoc.org/repos/omdoc/trunk/doc/primer/main.pdf`. 2009. URL: `https://svn.omdoc.org/repos/omdoc/trunk/doc/primer/main.pdf` (cit. on p. 17).

[Koh09b] Michael Kohlhase. “OMDoc Projects and Applications [Version 1.6 (pre-2.0)]”. Draft `https://svn.omdoc.org/repos/omdoc/trunk/doc/projects/main.pdf`. 2009. URL: `https://svn.omdoc.org/repos/omdoc/trunk/doc/projects/main.pdf`.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer, 1993.

[LS99] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. World Wide Web Consortium (W3C), 1999. URL: `http://www.w3.org/TR/1999/REC-rdf-syntax`.

[MAH06] Till Mossakowski, Serge Autexier, and Dieter Hutter. “Development Graphs – Proof Management for Structured Specifications”. In: *Journal of Logic and Algebraic Programming* 67.1–2 (2006), pp. 114–145.

[Mel+03] Erica Melis et al. “Knowledge Representation and Management in ActiveMath”. In: *Annals of Mathematics and Artificial Intelligence* 38.1–3 (2003), pp. 47–64 (cit. on p. 5).

[Mos04] P. D. Mosses, ed. Casl *Reference Manual*. LNCS 2960 (IFIP Series). Springer Verlag, 2004.

[MR03] *MARC code list for Relators, Sources, Description Conventions*. 2003. URL: `http://www.loc.gov/marc/relators`.

[MSK01] M. Murata, S. St. Laurent, and D. Kohn. *XML Media Types*. RFC 3023. Jan. 2001. URL: `ftp://ftp.isi.edu/in-notes/rfc3023.txt`.

[Mur+] Peter Murray-Rust et al. *Chemical Markup Language (CML)*. `http://cml.sourceforge.net/`. seen January 2007. URL: `http://cml.sourceforge.net/`.

[MVW05] Jonathan Marsh, Daniel Veillard, and Norman Walsh. `xml:id` *Version 1.0*. Tech. rep. World Wide Web Consortium (W3C), Sept. 9, 2005. URL: `http://www.w3.org/TR/2005/REC-xml-id-20050909/` (cit. on p. 8).

[NS81] Alan Newell and Herbert A. Simon. "Computer Science as empirical inquiry: Symbols and search". In: *Communications of the Association for Computing Machinery* 19 (1981), pp. 113–126.

[OMDoc] Michael Kohlhase. OMDoc*: An open markup format for mathematical documents (latest released version)*. Specification, `http://www.omdoc.org/pubs/spec.pdf`. URL: `http://www.omdoc.org/pubs/spec.pdf` (cit. on p. 6).

[Pau94] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS 828. Springer Verlag, 1994.

[PO8888] *IEEE POSIX*. ISO/IEC 9945. 1988.

[Sac06] Claudio Sacerdoti Coen. "Explanation in Natural Language of $\overline{\lambda}\mu\overline{\mu}$-terms". In: *Mathematical Knowledge Management, MKM'05*. Ed. by Michael Kohlhase. LNAI 3863. Springer Verlag, 2006.

[Sie+00] Jörg Siekmann et al. "Adaptive Course Generation and Presentation". In: *Proceedings of ITS-2000 workshop on Adaptive and Intelligent Web-Based Education Systems*. Ed. by P. Brusilovski and Chrisoph Peylo. Montreal, 2000.

[SZS04] G. Sutcliffe, J. Zimmer, and S. Schulz. "TSTP Data-Exchange Formats for Automated Theorem Proving Tools". In: *Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems*. Ed. by W. Zhang and V. Sorge. Frontiers in Artificial Intelligence and Applications 112. IOS Press, 2004, pp. 201–215.

[The02] The W3C HTML Working Group. *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0*. W3C Recommendation. World Wide Web Consortium (W3C), Aug. 1, 2002. URL: `http://www.w3.org/TR/2002/REC-xhtml1-20020801`.

[Tho91] Simon Thompson. *Type Theory and Functional Programming*. International Computer Science Series. Addison-Wesley, 1991.

[TLD] *Root-Zone Whois Information*. URL: `http://www.iana.org/cctld/cctld-whois.htm`.

[WM99] Norman Walsh and Leonard Muellner. *DocBook: The Definitive Guide*. O'Reilly, 1999.