# A Content Dictionary for In-Object Comments

Lars Hellström(✉)

Division of Applied Mathematics, The School of Education, Culture and Communication, Mälardalen University, Box 883, 721 23 Västerås, Sweden
lars.hellstrom@mdh.se

**Abstract.** It is observed that some OpenMath objects may benefit from containing comments. A content dictionary with suitable attribution symbols is proposed. This content dictionary also provides application symbols for constructing comments that are somewhat more than just plain text strings.

## 1 Introduction

OpenMath [1], of which is Content MathML [6, Ch. 4] is one encoding, is the open standard for machine-readable formalised mathematics. Though an early motivating application was the exchange of formulae between different Computer Algebra Systems, the standard is by means restricted to that context; other applications include the formalised mathematics aspect of documents written in the Lurch [2] word processor and semantic export of material [5] from the Digital Library of Mathematical Functions (DLMF) [3].

One key ingredient in the standard is that the many symbols needed to express mathematical claims as object are not defined in the standard itself, but by separate documents called Content Dictionaries (CDs) that may be created by anyone; ultimately each symbol is uniquely identified by a URI, and a web browser might use this to fetch the defining CD if that is made available by an appropriately located and configured server. The content dictionaries are also where OpenMath (OM) gets a bit recursive, since CDs typically contain OM objects (formulae) stating Formal Mathematical Properties (FMPs) of the symbols being defined. Though not *necessarily* the kind of rigorous axiomatisations one would expect in an automated theorem prover system, FMPs have the potential of fulfilling that role, and as such they may sometimes need to be rather large. Sheer size can then make it difficult for a reader to understand what is being stated (e.g., what would happen in a particular edge case), even when being well versed in the mathematics as such. This phenomenon of unavoidable complexity making things difficult to understand is utterly familiar from the realm of programming, and there it has a time-tested solution: document your code, and in particular *include appropriate comments* in it!

Being an application of XML, the OpenMath CD format of course allows XML comments, but those would only benefit a person editing the CD, which is


A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 473–481, 2020.
https://doi.org/10.1007/978-3-030-52200-1_47

a minor activity. More common is to use the CD as a reference or even authority on the meanings of symbols, and what that kind of document needs in order to be more easily digested are *annotations*—comments that are visible to the reader of a presentation of the document. Since the normative form of the CD will go through XML processing when generating such a presentation, XML comments are no good. Luckily the OpenMath standard already contains a feature that suits this end perfectly: attributions. One may to any OpenMath element (subformula) attach anything, and as long as the attribution symbol is not of the semantic variety, any OpenMath phrasebook (processor of OM data) is free to ignore it.

What prevents authors of CDs, and other OM objects of a more persistent nature, from already including annotations in them is that there has not existed any CD with appropriate symbols (they need to be declared as having the Role of 'attribution') to use as keys in these attributions. That is of course easily remedied by creating such a CD, and the primary contribution in this work. Secondarily, this work also provides some additional symbols that may be used to provide simple markup of the comments; the rationale behind this is discussed below, in conjunction with the actual symbol definitions.

One example below shows the OM-XML encoding of a comment attribution, for maximal clarity, but most of them employ a more compact "semiformula" format. In those examples, @ denotes application (OMA), **attr** denotes attribution (OMATTR), **bind** denotes binding (OMBIND), and symbols are written as *cd.name* (e.g. 'relation1.eq' for the standard equality relation =). Outside of semiformulae, references to symbols are rather written on the URI form *cd*`#`*name*, e.g. `relation1#eq`.

## 2 The `comment1` Content Dictionary

**Description:** Symbols to attach comments as attributions of OpenMath objects.
Standard OpenMath licence terms apply.

*Note 1.* Due to the size constraints of the ICMS proceedings, some material (including entire symbol definitions) have been omitted from this presentation. On the other hand, this section is based on the literate source for the content dictionary.

### 2.1 Basics

The purpose of attaching comments to objects may need an explanation, since not all kinds of comments make sense in all OpenMath usage scenarios. One factor of importance is the lifetime of the objects in question; objects that only stick around for a short time will at most have comments that are generated automatically (perhaps inserted by tools examining OpenMath objects, as a means of reporting on their findings), but objects of a very long duration (such

as those in Formal Mathematical Properties) could well acquire manually written comments as part of content dictionary maintenance.

**Symbol** `remark` (attribution)

| |
|---|
| The generic (neutral) comment attribution symbol. |

*Commented Mathematical Property.* As an attribution, the value may be data in some foreign format, but expressing comments as OpenMath objects can be more to the point as well as easier on implementors (since any tool working with comment attributions must already be able to read and/or write OM). In particular, a comment that is a piece of plain text can be succintly encoded as an OpenMath string.

**Example.**

```
<OMOBJ>
  <OMA> <OMS cd="arith1" name="divide"/>
    <OMATTR>
      <OMATP>
        <OMS cd="comment1" name="remark"/>
        <OMSTR>This cancellation may degrade numeric precision.</OMSTR>
      </OMATP>
      <OMA> <OMS cd="arith1" name="minus"/>
       <OMI> 1 </OMI>
       <OMA> <OMS cd="transc1" name="cos"/>
         <OMV name="x"/>
       </OMA>
     </OMA>
   </OMATTR>
   <OMA> <OMS cd="transc1" name="sin"/>
     <OMV name="x"/>
   </OMA>
 </OMA>
</OMOBJ>
```

In particular, using instead e.g. XHTML+MathML as format for comments, one would create an "industry standard" situation rather than a "math in the middle" situation: it would be easy on tools that aim to output HTML or the like, but very demanding for tools that do not target HTML.

Comments are primarily aimed at human readers, so the tools that take notice of comments would normally be those that generate a presentation. Even in that context, comments need not have much visible effect; in an interactive medium (e.g. a web page) it might be most appropriate to leave comments invisible until the user undertakes some action to display them (such as placing the cursor on top of the commented object, for displaying the comment as a tooltip).

One type of comment that might require visibility by default are headings, i.e., comments that aim to help a reader discern the essential structure of an OM object. In normal text, headings present little trouble as they apply at distinct

vertical positions and may be inserted between paragraphs or in the margin, but an OM object presented as a standard typeset formula may well warrant several headings on a single line, and then the problem of what falls under which heading becomes more troublesome. (One approach worth exploring could be to tint the background according to the heading which applies.) Generating presentations is not the issue here, but providing the information required to do so is, and just saying "heading" risks creating an ambiguity regarding the commenter's intents.

The problem is that text headings are normally understood as applying to all text up to the next (same level) heading, and someone editing an OM object in e.g. OM-XML format is likely to apply that interpretation, but OM objects grammatically rather have a tree structure, so phrasebook authors are likely to instead interpret a heading as applying only to the explicitly attributed element. In order to curtail that ambiguity, this CD has two heading symbols that explicitly state which kind of scope is intended.

**Symbol** `s-heading` (attribution)

This symbol attaches a comment to an object, classifying that comment as a heading. The scope of that heading consists of the attributed element and all its later siblings (up to the next sibling with an s-heading attribution, if any).

**Example.** While giving a long list of conditions, an author will often spontaneously organise these into blocks, but is perhaps not prepared to let those blocks be explicit in the structure of the formula; authors may be more willing to write

@(logic1.and, **attr**(comment1.s-heading "`First block`", *c1* ), *c2 , c3 , c4 , c5,*
**attr**(comment1.s-heading "`Second block`", *c6* ), *c7 , c8*)

than the logically equivalent

@(logic1.and, **attr**(comment1.t-heading "`First block`",
@(logic1.and, *c1 , c2 , c3 , c4 , c5* )),
**attr**(comment1.t-heading "`Second block`", @(logic1.and, *c6 , c7 , c8* )))

**Symbol** `t-heading` (attribution)

This symbol attaches a comment to an object, classifying that comment as a heading. The scope of that heading is exactly the element to which the comment is attached, i.e., it adheres strictly to the tree structure of the OM object.

**Example.** The distinction between left and right hand side of an implication is explicit in the tree structure of the formula, but inserting explicit headings can still improve the overall human readability.

**bind**(quant1.forall; *desc*, *p*, *epsilon*;
@(logic1.implies; **attr**(comment1.t-heading "`Conditions`",
@(logic1.and, @(Riemann-surface.is_description, *desc*),
@(set1.in, *p*, @(comment1.set, *desc*)),
@(set1.in, *epsilon*, setname1.R), @(relation1.gt, *epsilon*; 0))),
**attr**(comment1.t-heading "`Claims`";
@(logic1.and, @(set1.prsubset, @(set1.set, *p*),
@(Riemann-surface.neighbourhood, *desc*, *p*, *epsilon*)),
@(set1.subset, @(Riemann-surface.neighbourhood, *desc*, *p*, *epsilon*),
@(Riemann-surface.set, *desc*))))))

**Symbol** `question` (attribution)

This symbol attaches a comment to an object, classifying that comment as a question.

**Example**

**attr**(comment1.question
"`Is this formalisation what we want also under intuitionistic logic?`",
@(logic1.implies, @(logic1.not, $Q$), @(logic1.not, $P$)))

*Commented Mathematical Property.* The extra signal sent by classifying a comment as a question is that an answer to it is sought. Presentation-wise, it might not be much different from a plain remark, but if one has a large collection of OM objects then the ability to easily search for those with pending questions is obviously valuable.

**Symbol** `warning` (attribution)

This symbol attaches a comment to an object, classifying that comment as a warning.

**Symbol** `diagnostic` (attribution)

This symbol attaches a comment to an object, classifying that comment as containing diagnostic information.

### 2.2 Comment Format Functions

A human reader can usually guess what format a piece of text is, but automated processing is much aided by allowing material of a specific form to be marked up as such.

**Symbol** `uri` (application)

> This function expects a string as argument, and asserts that this string is a Uniform (or more generally Internationalized) Resource Identifier, producing a corresponding comment object as result.

**Example.** Does fns1#range denote the image of a function or the codomain of a function? There is a tracker issue on this.

@(**attr**(comment1.question
@(comment1.uri, "`https://github.com/OpenMath/CDs/issues/4`"),
fns1.range), $f$)

**Symbol** `formula` (application)

> This function can take an arbitrary OpenMath object as argument, and asserts that the comment is this object as a formula, rather than the interpretation of that object. To the author, this is essentially to enter "math mode".

**Example.** An expression ln(-x) strikes most readers as strange, but in the case that $x < 0$ there is of course nothing strange or problematic about it at all.

**attr**(comment1.remark@(comment1.formula, @(relation1.lt, $x$, 0)),
@(transc1.ln, @(arith1.unary_minus, $x$)))

A content dictionary only specifies what an OM object means, not how it is to be processed, but a reasonable mode of processing comments is to see each function as returning a presentation (in some suitable form) of the comment in question; this is feasible because the vocabulary of "comment-valued" applications is fairly small. However if a general formula is to be used as a comment then this may obviously contain arbitrary symbols and constructions, so the formula symbol serves as a sign to any processing phrasebook that a different mode of processing should be applied for this subobject.

The `meta#Example` symbol arguably provides an imperfect encoding of OM CD Examples because it lacks this kind of symbol: it cannot distinguish between a string that is text and a string that is an embedded OMOBJ that is just a string, whereas the standard encoding of that content dictionary document would make this distinction.

*Commented Mathematical Property.* It is a feature of formulae in comments that they may contain variables whose scopes extend outside the comment. Renaming a variable through alpha-conversion then causes not only occurrences of that variable in the main OMOBJ to be renamed, but also occurrences in comment

attributions in that OMOBJ, and this happens as a consequence of the general principles in the OM standard. Foreign encodings of comments would not be so lucky.

**Symbol** `mixed` (application)

| This function concatenates an arbitrary number of arbitrary comment pieces into one comment. |
|---|

This is named for the `multipart/mixed` content-type in MIME.

*Commented Mathematical Property.* Heuristics may be applied when joining different pieces together, but expect there to be interword spaces between the pieces when concatenated.

**Example**

**attr**(comment1.remark
@(comment1.mixed, "`This comment is`", "`split over two strings.`"),
@(relation1.eq, @(arith1.plus, 1, 1), 2))

### 2.3 Comment Structuring Functions

It can sometimes be useful to attach metadata to comments. The following are some function symbols which take a comment as their first argument, producing an embellished comment as result.

**Symbol** `when` (application)

| This function attaches a timestamp to a comment. |
|---|

*Commented Mathematical Property.* The first argument is the comment to act upon, and the second argument is the timestamp to attach. If the second argument is a number, then that is interpreted as the number of seconds since the new year 1970 CE (the UNIX epoch). The second argument may alternatively be a string expressing the date; the format of that string is then not mandated, but ISO 8601 is recommended.

**Symbol** `language` (application)

| This function specifies the language used in a comment. |
|---|

*Commented Mathematical Property.* The first argument is the comment. The second argument is an IETF language tag.

**Symbol `alternatives`** (application)

| |
|---|
| This function takes an arbitrary number of comments as arguments, making the assertion that they all say the same thing (although probably in different ways) and return a combined comment where the individual comments are distinct pieces. |

**Example.** This can be combined with the language symbol to collect different language versions of the same comment.

**attr**(comment1.remark
  @(comment1.alternatives, @(comment1.language, "`continuous`", "`en`"),
    @(comment1.language, "`stetig`", "`de`")), $f$)

**Symbol `signed`** (application)

| |
|---|
| This function attaches a signature to a comment. |

*Commented Mathematical Property.* The first argument is the comment. The second argument identifies the entity (not necessarily a person) signing the comment.

*Commented Mathematical Property.* If there are additional arguments, then the third argument names a digital signature scheme and the fourth is a signature under that scheme. (This CD does not define any signature schemes.)

On the other hand, for example RFC5652 [4] does define signature schemes. What those sign are octet-sequences, so one need only define a way of encoding the OpenMath objects being signed as such. One obvious possibility is to use the binary encoding of the material signed, normalised to make this encoding canonical (expanding all references and using the shortest possible encoding form should suffice quite far).

## References

1. Buswell, S., et al.: The openmath standard. Technical report, The OpenMath Society, July 2019. https://www.openmath.org/standard/om20-2019-07-01/
2. Carter, N.C., Monks, K.G.: Lurch: a word processor that can grade students' proofs. In: Lange, C., et al. (eds.) MathUI, OpenMath, PLMMS and ThEdu Workshops and Work in Progress at the Conference on Intelligent Computer Mathematics. No. 1010 in CEUR Workshop Proceedings, Aachen (2013). http://ceur-ws.org/Vol-1010/paper-04.pdf
3. Olver, F.W.J., et al.: NIST Digital Library of Mathematical Functions. Release 1.0.26 of 2020-03-15. http://dlmf.nist.gov/
4. Housley, R.: Cryptographic Message Syntax (CMS). RFC 5652 (Internet Standard), September 2009. https://doi.org/10.17487/RFC5652. https://www.rfc-editor.org/rfc/rfc5652.txt

5. Rabe, F., Farmer, W.M., Passmore, G.O., Youssef, A. (eds.): CICM 2018. LNCS (LNAI), vol. 11006. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96812-4
6. Miner, R.R., Ion, P.D.F., Carlisle, D.: Mathematical markup language (MathML) version 3.0, 2nd edn. W3C recommendation, W3C, April 2014. http://www.w3.org/TR/2014/REC-MathML3-20140410/