

OMDoc Theory Graphs Revisited

Florian Rabe

Jacobs University Bremen

Abstract. We propose extensions and corrections of the syntax and semantics of OMDoc that are necessary to define a formal semantics of OMDoc theory graphs. Since OMDoc theories are also OPENMATH content dictionaries, this provides a module system and rudimentary syntax checking for OPENMATH. Our proposal includes a new constructor for OPENMATH objects that is needed in the context of this module system. Together with related work, this contribution provides a formal semantics and full type checking for OMDoc theory graphs.

1 Introduction

OMDoc ([Koh06]) provides an XML language for the specification of mathematical theories. An OMDoc theory consists of a list of subtheories, symbols, possibly with types and definitions, axioms, and imports, the latter providing a generic module system. But OMDoc 1.2 has two serious shortcomings.

Firstly, a theory T cannot import a theory S in two different ways. This situation arises, for example, when constructing the theory of commutative groups from the theory of monoids, and then importing both to form the theory of rings, which produces a theory graph with two different imports from monoid to ring. Instead such imports are implicitly identified.¹

And secondly, logical theories are not absolute but are relative to a logic. While it is natural to assume that the logic is represented by an OMDoc theory as well, this connection is not considered explicitly.

Both these problems are negligible for certain mathematical systems, namely those which are used for concrete mathematics, i.e., analysis or number theory where the dictionary of mathematical symbols is fixed. Therefore, OMDoc can be used naturally as a content dictionary architecture for OPENMATH ([BCC⁺]), and the specific theories that are part of the OPENMATH canon have a semantics by informally relating the declared symbols to the mathematical objects they represent. But for abstract mathematics, e.g., algebraic specification or model

¹ In fact, the OMDoc specification does mention a way to import the same theory in two different ways, see the example in Sect. 18.1. However, this is more like a hack than a clean solution and leads to several problems. For example, it becomes necessary to require that both imports leave disjoint sets of symbols of the imported theory unmapped, which leads to the artifact of having to declare the symbols "times" and "one" in List. 18.4. Furthermore, the doubly imported axioms cannot be distinguished even though they give rise to non-equivalent axioms in the importing theory.

theory, these shortcomings render OMDoc unusable. Indeed, there is no complete formal or even informal semantics for OMDoc theories.

Our work was motivated by the desire to provide such a general semantics, and this required several improvements and corrections to the OMDoc specification. In the following we will describe the syntax and semantics of the relevant parts of OMDoc as we suggest it for the upcoming OMDoc 2 standard. We will mention the most important changes by footnotes.

2 Theory Graphs

By a *theory graph*, we mean an OMDoc document containing theory and theory-inclusion elements. Objects over a theory represent terms, types, etc.

2.1 Valid Objects

O is a *valid object*² over T if it is a valid object over T and the empty set of variables. And valid objects over T and the set of variables Var are defined as follows:³

- If n is a symbol (see below) of T , then $\langle \text{OMS cd}="T" \text{ name}="n" / \rangle$ is valid over T and Var .
- If $V \in Var$, V is valid over T and Var .
- If O is valid over the meta-theory of T and Var , it is valid over T and Var .
- If O and O_1, \dots, O_n are valid over T and Var , then so are application and the error of O with arguments O_1, \dots, O_n written $\langle \text{OMA} \rangle O O_1 \dots O_n \langle / \text{OMA} \rangle$ and $\langle \text{OME} \rangle O O_1 \dots O_n \langle / \text{OME} \rangle$.
- If O_1 is valid over T and Var , V_i are possibly attributed variables, and O_2 is valid over $Var \cup \{V_1, \dots, V_n\}$, then the binding $\langle \text{OMBIND} \rangle O_1 \langle \text{OMBVAR} \rangle V_1 \dots V_n \langle / \text{OMBVAR} \rangle O_2 \langle / \text{OMBIND} \rangle$ is valid over T and Var .
We will consider, e.g., $\langle \text{OMBIND} \rangle O \langle \text{OMBVAR} \rangle V_1 V_2 \langle / \text{OMBVAR} \rangle O' \langle / \text{OMBIND} \rangle$ as an abbreviation of $\langle \text{OMBIND} \rangle O \langle \text{OMBVAR} \rangle V_1 \langle / \text{OMBVAR} \rangle \langle \text{OMBIND} \rangle O \langle \text{OMBVAR} \rangle V_2 \langle / \text{OMBVAR} \rangle O' \langle / \text{OMBIND} \rangle \langle / \text{OMBIND} \rangle$.
- If $\langle \text{OMS cd}="S" \text{ name}="key" / \rangle$ is valid over T and Var , and if Val and O are valid over T and Var , then so is the attribution $\langle \text{OMATTR} \rangle \langle \text{OMATP} \rangle \langle \text{OMS cd}="S" \text{ name}="key" / \rangle Val \langle / \text{OMATP} \rangle O' \langle / \text{OMATTR} \rangle$.
We will consider, e.g., $\langle \text{OMATTR} \rangle \langle \text{OMATP} \rangle K_1 Val_1 \langle / \text{OMATP} \rangle \langle \text{OMATP} \rangle K_2 Val_2 \langle / \text{OMATP} \rangle O \langle / \text{OMATTR} \rangle$ as an abbreviation of $\langle \text{OMATTR} \rangle \langle \text{OMATP} \rangle K_2 Val_2 \langle / \text{OMATP} \rangle \langle \text{OMATTR} \rangle \langle \text{OMATP} \rangle K_1 Val_1 \langle / \text{OMATP} \rangle O \langle / \text{OMATTR} \rangle \langle / \text{OMATTR} \rangle$.
- If O is valid over S and $\langle \text{OMM via}="via" / \rangle$ is a theory morphism (see below) from S to T , then $\langle \text{OMM via}="via" \rangle O \langle / \text{OMM} \rangle$ is valid over T .⁴

An *elementary object* is a symbol or of the form $\langle \text{OMM via}="imp" \rangle O \langle / \text{OMM} \rangle$ for an imports (see below) *imp* and an elementary object O .

² Change: The OMDoc 1.2 semantics does not have the concept of validity for objects.

³ We do not consider content MATHML.

⁴ Change: The OMDoc 1.2 syntax does not have OMM elements. This is insufficient since it is impossible to refer to an imported symbol.

2.2 Theories

A *theory* T has the form^{5 6}

```
<theory xml:id="T" meta="M">
  symbols, definitions, axioms, imports
</theory>
```

where the optional M refers to another theory, called the *meta-theory* of T . In the following let T be a fixed theory.

A symbol has the form⁷:

```
<symbol name="n">
  <type>
    <OMOBJ>O</OMOBJ>
  </type>
</symbol>
```

where O must be a valid object over T . We call O the type of n .

A simple⁸ definition has the form:

```
<definition for="#n">
  <OMOBJ>O</OMOBJ>
</definition>
```

where n must be a symbol name in T and O must be valid over T . We call O the definiens of the symbol n .

An axiom has the form:⁹¹⁰

```
<axiom name="n">
  <FMP via="via"><OMOBJ>O</OMOBJ></FMP>
</axiom>
```

where *via* is a theory morphism from some S to T and O is valid over S .

An *import* *imp* from the theory S (into T) has the following form¹¹

```
<imports name="imp" from="S" meta="m" type="t">
  <morphism>map* </morphism>
</imports>
```

where m is a theory morphism from the meta-theory of S into the meta-theory of T , called the meta-morphism of *imp*. (m is omitted if S has no meta-theory.)

⁵ Change: The OMDoc 1.2 syntax does not have the concept of meta-theories.

⁶ We do not treat subtheories yet.

⁷ We will not treat multiple types.

⁸ We will not treat the more complex definition types.

⁹ Change: The OMDoc 1.2 syntax has an attribute logic instead of our *via*. Its value corresponds to the domain of the theory morphism *via*, but that is insufficient since there may be more than one theory morphism between two theories.

¹⁰ For convenience, we use the attribute name for axioms and imports instead of `xml:id`, but in this work we do not treat the corresponding changes to the OMDoc syntax.

¹¹ Change: The OMDoc 1.2 syntax does not have the concept of meta-morphisms.

If the two meta-theories are equal and m omitted, the identity morphism is assumed.

map^* abbreviates a sequence of elements of the form

```
<requation>
  <OMOB>O</OMOB>
  <OMOB>O'</OMOB>
</requation>
```

for valid elementary objects O over S , and a valid object over T or theory morphism into T O' , and we say that imp maps O to O' . We require that O' is a theory morphism iff O is; and if so, that they have the same domain. The intuition is that symbols O in the domain of the map are instantiated with their images O' whereas the unmapped symbols of S are imported into T as new symbols.

If imp maps an import imp' into S to some import into T , this is equivalent to mapping every symbol in the domain of imp' separately; alternatively, this can be seen as an (asymmetric) sharing specification. This may lead to conflicting maps, namely if there are maps for both $\langle OMM\ via="via">s</OMM\rangle$ (where s may be empty) and $\langle OMM\ via="via'"/>$ such that via' is a suffix of via . We permit this case and define that the former, i.e., the more specific map, takes precedence.

An imports also imports the axioms of S into T . If t is "local", it imports all axioms locally declared in S . If t is "global", which is the default, it imports the imported axioms, too.¹²

2.3 Theory Morphisms

A *theory-inclusion* inc from S to S' is of the form

```
<theory-inclusion xml:id="inc" from="S" to="S'" meta="m"
  type="t">
  <morphism>...</morphism>
  <obligation induced-by="ax" assertion="ass"/>*
</theory-inclusion>
```

where m is as for imports. The morphism child of a theory-inclusion element must be as for an imports element except that it must map every (local or imported) symbol of S .¹³

A theory-inclusion inc represents a formal theory morphism: For every axiom ax of S , $\langle OMM\ via="inc">ax</OMM\rangle$ must be a theorem of S' . If t is "local", obligation elements must exist for all local axioms of S , if it is "global", which is the default, also for the imported axioms of S . obligation elements connect these axioms with the corresponding assertions.

Theory morphisms are defined by:

¹² Change: The OMDoc 1.2 semantics defines the semantics of local imports as importing neither imported axioms nor imported symbols. However, it is more reasonable to import all symbols.

¹³ We do not treat decompositions yet.

- $\langle \text{OMM via}="" \rangle / \rangle$ is a theory morphism from any theory into itself.
- If M is a theory morphism from R into S , and m refers to an imports or a theory-inclusion from S into T , then $\langle \text{OMM via}="" m \rangle M \langle / \text{OMM} \rangle$ is a theory morphism from R into T .
We will consider, e.g., $\langle \text{OMM via}="" m_1 m_2 \rangle / \rangle$ to be an abbreviation of $\langle \text{OMM via}="" m_2 \rangle \langle \text{OMM via}="" m_1 \rangle / \rangle \langle / \text{OMM} \rangle$.)

2.4 Normalization

Every theory morphism $\langle \text{OMM via}="" via \rangle / \rangle$ from S to T induces a mapping which maps a valid object O over S to the valid object $\langle \text{OMM via}="" via \rangle O \langle / \text{OMM} \rangle$ over T . This map is the composition of the natural extensions of the maps referenced by the imports or theory-inclusion elements in via .

Normalization eliminates some OMM elements by the following prioritized rewrite rules. The idea is to push all theory morphisms to the inside of a term (i.e., from the root to the leaves of its syntax tree), and then to apply the morphism if it hits a symbol for which a map id defined.

1. Expand all abbreviations, i.e., of attributions, bindings, and morphisms.
2. Rewrite $\langle \text{OMM via}="" m \rangle O \langle / \text{OMM} \rangle$ for an application, error, or binding O by pushing the morphism to the inside. For example, rewrite $\langle \text{OMM via}="" m \rangle \langle \text{OMBIND} \rangle O \langle \text{OMBVAR} \rangle V \langle / \text{OMBVAR} \rangle O' \langle / \text{OMBIND} \rangle \langle / \text{OMM} \rangle$ to $\langle \text{OMBIND} \rangle \langle \text{OMM via}="" m \rangle O \langle / \text{OMM} \rangle \langle \text{OMBVAR} \rangle \langle \text{OMM via}="" m \rangle V \langle / \text{OMM} \rangle \langle / \text{OMBVAR} \rangle \langle \text{OMM via}="" m \rangle O' \langle / \text{OMM} \rangle \langle / \text{OMBIND} \rangle$.
3. If m refers to an imports or a theory-inclusion element mapping the symbol K to the symbol K' , rewrite $\langle \text{OMM via}="" m \rangle \langle \text{OMATTP} \rangle \langle \text{OMATTP} \rangle K V \langle / \text{OMATTP} \rangle O \langle / \text{OMATTP} \rangle \langle / \text{OMM} \rangle$ to $\langle \text{OMATTP} \rangle \langle \text{OMATTP} \rangle K' \langle \text{OMM via}="" m \rangle V \langle / \text{OMM} \rangle \langle / \text{OMATTP} \rangle O \langle / \text{OMATTP} \rangle$. Otherwise, signal an error.
4. Rewrite $\langle \text{OMM via}="" \rangle \langle \text{OMV name}="" v \rangle / \rangle \langle / \text{OMM} \rangle$ to $\langle \text{OMV name}="" v \rangle / \rangle$.
5. If m refers to an imports or a theory-inclusion element from S to T with meta-morphism m' , and $S' \neq S$, rewrite $\langle \text{OMM via}="" m \rangle \langle \text{OMS cd}="" S' \rangle \text{name}="" n \rangle / \rangle \langle / \text{OMM} \rangle$ to $\langle \text{OMM via}="" m' \rangle \langle \text{OMS cd}="" S' \rangle \text{name}="" n \rangle / \rangle \langle / \text{OMM} \rangle$.
6. Rewrite $\langle \text{OMM via}="" \rangle O \langle / \text{OMM} \rangle$ to O .
7. If m refers to an imports or a theory-inclusion element mapping O to O' , rewrite $\langle \text{OMM via}="" m \rangle O \langle / \text{OMM} \rangle$ to O' .
8. If m refers to an imports or a theory-inclusion element mapping the theory morphism $\langle \text{OMM via}="" m_1 \dots m_i \rangle / \rangle$ to $\langle \text{OMM via}="" via \rangle / \rangle$ but not mapping any more specific theory morphism, rewrite $\langle \text{OMM via}="" m \rangle \langle \text{OMM via}="" m_1 \rangle \dots \langle \text{OMM via}="" m_n \rangle O \langle / \text{OMM} \rangle \dots \langle / \text{OMM} \rangle \langle / \text{OMM} \rangle$ to $\langle \text{OMM via}="" via \rangle \langle \text{OMM via}="" m_{i+1} \rangle \dots \langle \text{OMM via}="" m_n \rangle O \langle / \text{OMM} \rangle \dots \langle / \text{OMM} \rangle \langle / \text{OMM} \rangle$.
9. Reintroduce the abbreviations of theory morphisms.

This will not eliminate all OMM elements. For unmapped imported symbols S , objects of the form `<OMM via="imp1">S</OMM>` will remain. And if imp_2 maps neither `<OMM via="imp1">S</OMM>` nor `<OMM via="imp1"/>`, objects of the form `<OMM via="imp1 imp2">S</OMM>` will remain, and so on. In other words, `<OMM via="imp1 ... impn">S</OMM>` is the normal form of free imported symbols, i.e., imported symbols that are not coerced to be equal to some object due to a map. (Of course, they may be coerced arbitrarily by imported or locally declared axioms.)

It is straightforward to show that this is a terminating and normalizing rewrite system. Its intuitive meaning is the following: If there are no definitions and no axioms, two objects are semantically equal iff they have the same normal form.

2.5 Assertions and proofs

Assertions correspond to axioms and have the form

```
<assertion xml:id="ass" theory="T">
  <FMP via="via">A</FMP>
</assertion>
```

where T refers to the theory in which the assertion holds.

Proofs in the theory T are of the form^{14 15}

```
<proof xml:id="p" for="ass">
  <hypothesis name="H"><FMP>O</FMP></hypothesis>*
  derive steps
</proof>
```

and derive steps are again proof elements or are of the form

```
<derive xml:id="id" type="t">
  <method>M</method>
  <parameter>P</parameter>*
  <premise xref="ref"/>*
  <derivate><FMP>F</FMP></derivate>
</derive>
```

where ass refers to an assertion over T . Proof steps given by proof elements may also occur at top-level. t may be "step", which is the default, "conclusion", "gap", or "translation". Giving the derivate child is optional except for steps of the type "gap".

A proof represents a formal proof tree with derive steps as nodes. The root is given by a unique derive step with the type "conclusion". The children of a node

¹⁴ Change: Our changes to the syntax of proofs are of cosmetic nature except for the following. Firstly, the method used in a derive step explicitly refers to an object as the proof rule, which is not covered in OMDoc 1.2. And secondly, translation steps do not exist in OMDoc 1.2.

¹⁵ We do not treat symbols declared or defined locally in a proof yet.

D are given by the target S of the xref attributes of the premise elements of D : If S is a derive element, S is a child; if S is a proof element, its root is a child, i.e., S is a subtree. The leaves can be proof rules that do not take premises, references to axioms or assertions of the home theory, or to hypotheses of an enclosing proof element.

Within a derive step, M is a valid object over T giving the proof rule; typically, M will be valid over the meta-theory of T . P in a parameter must be a valid object over T giving a parameter for the rule application, and F is the derived formula. Alternatively, a derive step may have type "gap", in which case method and parameters must be omitted. And if the derive step has type "translation", M must be a theory-morphism from some S into T , parameters must be omitted, and the only premise must refer to a proof over S of some A , such that F is the result of translating A via M .

3 Valid Theory Graphs and Logical Semantics

Finally, the graph induced by the references between any elements must be acyclic. And two OMDoc documents that satisfy this requirement and that differ only in the order of the children of the top-level are considered semantically equal. Within a theory, the order of children is part of the definition of the theory except for definitions, which may occur anywhere. (When importing a theory, its order must be reserved.) In particular, we can assume without loss of generality that all the references in a theory graph go against the document order, and that all definitions in a theory occur at the end of the theory in arbitrary order.¹⁶

Note that this does not exclude mutual recursion: Only the types of two symbols may not be mutually recursive, but any definition may refer to any previously declared symbol. Thus two mutually recursive definitions can be given if both symbols are declared before both definitions are given.

However, it is not possible to define in general when a theory graph should be considered valid because the following questions regarding typing and equality checking cannot be answered on the OMDoc level.

- Which objects are eligible as the type and definiens of a symbol, or as axioms?
- Which definiens-type combinations are permitted?
- Which morphism elements preserve types? This depends on subtle typing conditions such as: If a sort a is mapped to a' , a constant of sort a must be mapped to a term of sort a' .
- When can an imports map an imported symbol? If imp maps $\langle OMM \text{ via}="imp_1 \dots imp_n">s\langle /OMM \rangle$ to O for some symbol s , and some imp_i already maps $\langle OMM \text{ via}="imp_1 \dots imp_{i-1}">s\langle /OMM \rangle$ to O' , then O must be equal to $\langle OMM \text{ via}="imp_{i+1} \dots imp_n imp">O'\langle /OMM \rangle$.
- When can an imports imp map a theory morphism m to m' ? If m maps O to O' , then $\langle OMM \text{ via}="imp">O'\langle /OMM \rangle$ must be equal to $\langle OMM \text{ via}="m'">O\langle /OMM \rangle$.

¹⁶ Change: OMDoc 1.2 is silent about whether the order of the children is relevant.

- Which proofs are correct?

The basic idea to give theories in OMDoc a formal semantics is given by the model-theoretic framework of institutions ([GB92]). Roughly, there must be one theory without (local or imported) axioms and without any imports containing morphism elements. For this theory, an institution must be given outside of OMDoc. It will serve as a logical framework answering the above questions. Then, for example, logics can be represented as theories with the logical framework as a meta-theory, and logical theories as theories with a logic as a meta-theory. In [Rab07], we described this for the case of LF ([HHP93]) as the logical framework. There OMDoc symbols and axioms are interpreted as LF-constants, objects and proofs as LF-terms, and FMP elements as LF-types accordings to the propositions as types-paradigm. Then well-typedness of the resulting LF signature defines validity of the OMDoc theory graph.

References

- [BCC⁺] S. Buswell, O. Caprotti, D. Carlisle, M. Dewar, M. Gaëtano, and M. Kohlhase. The openmath standard. See <http://www.openmath.org/cocoon/openmath/standard/>.
- [GB92] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.
- [HHP93] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.
- [Koh06] M. Kohlhase. *OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2)*. Number 4180 in Lecture Notes in Artificial Intelligence. Springer, 2006.
- [Rab07] F. Rabe. Interpreting Theory Graphs in a Logical Framework. Submitted, see http://kwarc.eecs.iu-bremen.de/frabe/Research/rabe_omdoclf_07.pdf, 2007.