OpenMath-RDF: RDF encodings for OpenMath objects and Content Dictionaries

Ken Wenzel

Fraunhofer-Institute for Machine Tools and Forming Technology IWU, Chemnitz, Germany

Abstract

This paper presents RDF encodings for OpenMath objects and Content Dictionaries that allow mathematical objects and symbol definitions to be first-class citizens on the Web of Data.

1. Introduction

In 2003 Marchiori outlined the idea and possible applications of representing mathematical expressions as part of the Semantic Web [1]. Advantages are seen in an RDF [2, 3] representation that enables the reuse of Semantic Web languages and tools to support functions like search, annotation or inference on mathematical knowledge. Basic ideas for a direct conversion of MathML to RDF without an explicit ontology are also given in the paper. Marchiori also names the possible computability as a "cool functionality" of mathematical formulas on the Semantic Web.

In 2011 Lange [4] worked on methods for the collaborative creation and exchange of semiformal mathematical content. The authors introduce the OMDoc ontology (Open Mathematical Documents) for the exchange of mathematical statements and theories on the internet. The ontology is defined in OMDoc itself since the authors state that the expressiveness of OWL is insuffient for the representation of all aspects of OMDoc [4, S. 102]. Additional to OMDoc an OWL based ontology for the description of OpenMath Content Dictionaries is introduced. It is able to represent metadata about symbols and their usage within mathematical expressions but not the expressions themselves. Both ontologies are used to implement a wiki system called SWiM (Semantic Wiki for Mathematical Knowledge Management) for the collaborative work on mathematical documents.

In 2012 Ferré [5] proposes a lightweight RDF vocabulary for the representation of mathematical expressions mainly for the use case of content-based search. The vocabulary is solely based on existing RDF and RDFS properties and hence there is no explicit ontology. The property `rdf:type` is used as constructor of mathematical operations where each object is an instance of `rdfs:Container` and the properties rdf:_1, rdf:_2, …, rdf:_n are used to represent its arguments.

31th OpenMath Workshop at CICM 2021, July 26–31, 2021 Timisoara, Romania

ken.wenzel@iwu.fraunhofer.de (K. Wenzel)

For example, the expression $(a + 2) * 3$ would be represented as

```
[ a math:Times ;
  rdf:_1 [ a math:Plus ; rdf:_1 _:a ; rdf:_2 2 ] ; rdf:_2 3 ] .
_:a rdfs:label "a" .
```

in Turtle [6] format.

The syntax is comparable to the notation used by the programming language Lisp that may represent the expression as

```
(math:Times (math:Plus a 2) 3)
```

Due to the missing ontology, semantics of the RDF representation is limited, for example, constants and variables are only implicitly distinguishable based on their node kind (constants are RDF literals and variables are blank nodes with a label). Since the representation was developed for structural search, a language construct for binding the variables of a lambda function as required for computations is not supported.

In 2014 Muñoz et al. 2014 [7] developed an ontology for mathematical expressions that also supports references from mathematical models to elements of a domain model. However, the approach uses a highly proprietary vocabulary and does not consider any standards like MathML or OpenMath.

This paper builds on the previous work of Wenzel and Reinhardt [8] that already introduced a basic ontology for OpenMath and a concept for mathematical computations on RDF graphs. This ontology is extended and the mapping between OpenMath-XML and OpenMath-RDF formalized. Furthermore an RDF encoding for Content Dictionaries is presented.

2. Ontology for OpenMath objects

The RDF data model for OpenMath objects can be specified as an OWL ontology as denoted by Figure 1.

The ontology was developed by starting with the OpenMath XML serialization and mapping of its elements like OMV or OMA to classes with speaking names like `Variable` or `Application` and by modelling their attributes as RDF properties.

2.1. Basic OpenMath objects

Integers, floating point numbers, character strings and byte arrays are directly represented using typed RDF literals. They are modelled by the class `:Literal` with the property `:value` for the associated values.

The ontology does not yet restrict the range of `:value`, and hence, besides the core OpenMath literal types `xsd:integer` (Integer), `xsd:double` (IEEE floating point number), `xsd:string` (Character string) and `xsd:base64Binary` (Byte array), allows other datatypes like `xsd:int` (32-bit integers) or `xsd:time`.

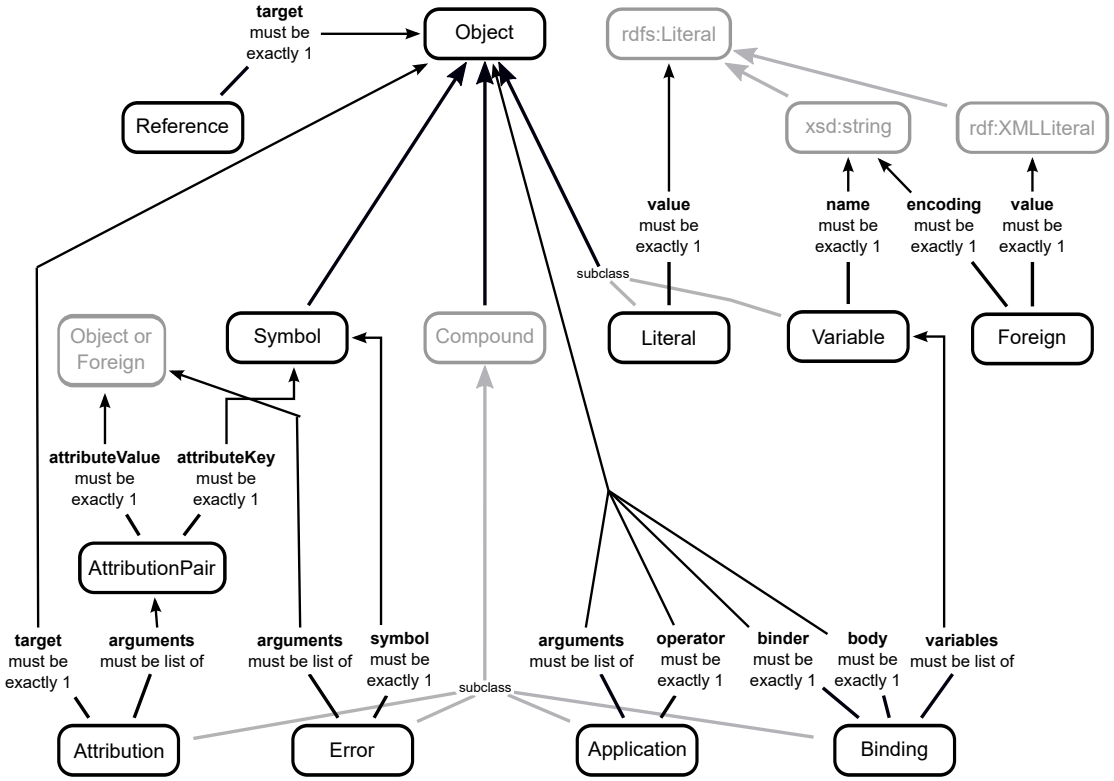


Figure 1: Ontology of OpenMath objects

For ensuring the compatibility with OpenMath 2 only the RDF types of core OpenMath literals should be used. To enforce this it is also possible to either restrict the range of `:value` by a union class (in Manchester OWL Syntax [9]) like

```
xsd:integer or xsd:double or xsd:string or xsd:base64Binary
```

or by defining subclasses of `:Literal` like `:IntegerLiteral` and others.

The OpenMath-RDF encoding also omits the hexadecimal encodings for numbers in favour of using standard RDF literal types. If support for hexadecimal literals is required then custom RDF datatypes can be used.

Variables are represented as instances of class `:Variable` with a property `:name`. The variable x can, for example, be represented as:

```
_:x a :Variable ; :name "x" .
```

It is allowed to freely choose the identification of RDF nodes in OpenMath-RDF as it has no influence of the semantics. Consequently, instead of using an anonymous RDF node `_:x` for the variable in the previous example, it could have been also named with a URI like `<var:x>` or `http://example.org/x` to allow global references.

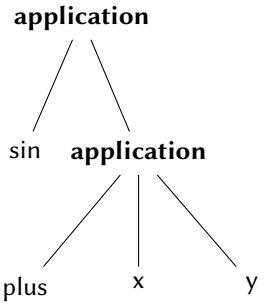
According to the OpenMath standard the scheme `<cdbase/cd#name>` [10, S. 16] can be used to create canonical URIs for symbols. Accordingly the symbol `sin` from the content

dictionary `transc1` can be encoded as `<http://www.openmath.org/cd/transc1#sin>`. OpenMath-RDF uses this kind of URIs to encode symbols and not the individual attributes `cdbase`, `cd` and `name` as used by the XML encoding.

2.2. Compound objects

A fundamental difference between XML and RDF lies in the fact, that XML uses a native tree structure while RDF is graph-based. For compound objects OpenMath-XML can rely on the tree structure to represent syntax trees of OpenMath expressions.

As an example Figure 2 shows on the left side the abstract syntax tree of the expression $sin(x+y)$ and on the right side the respective OpenMath-XML encoding. For the representation of application objects (OMA) in RDF the additional property `:arguments` is required. It references a list of arguments while the property `:operator` determines the function that is applied to to the arguments. A similar representation is also used for the other compound objects *binding* (OMBIND), *attribution* (OMATTR) and *error* (OME). This is in line with the OpenMath JSON encoding that also uses additional properties to represent parent-child relationships.




```
<OMA>
  <OMS cd="transc1" name="sin" />
  <OMA>
    <OMS cd="arith1" name="plus" />
    <OMV name="x" />
    <OMV name="y" />
  </OMA>
</OMA>
```

Figure 2: Syntax tree of the expression $sin(x + y)$ and its OpenMath-XML encoding

The resulting RDF representation of the expression is shown in Listing 1. Caused by the additional properties the syntax is less compact as with the XML format but offers stronger semantics. For example, the meaning of the child elements is directly expressed through the properties `:operator` and `:arguments`.

2.3. References

References are used within the XML and binary encodings to share OpenMath objects between multiple expressions. These references are basically not required in RDF since the linking of objects is directly supported through URIs or anonymous identifieres (blank nodes).

Nonetheless, OpenMath-RDF defines the class `:Reference` with the property `:target`. This opens up the possibilty to give OpenMath objects multiple names and to use the references as some kind of symbolic links as known by typical file systems.

Example 2.1 (Function with two different names). *The function $f(x) = x^2$ is mainly represented as RDF node with the URI* `<http://example.org/square-func>` *(Listing 2). By us-*

```
@prefix : <http://numerateweb.org/vocab/math#> .

[] a :Application ;
  :operator <http://www.openmath.org/cd/transc1#sin> ;
  :arguments (
    [ a :Application ;
      :operator <http://www.openmath.org/cd/arith1#plus> ;
      :arguments (
        [ a :Variable ; :name "x" ] [ a :Variable ; :name "y" ]
      )
    ]
  ) .
```

Listing 1: OpenMath-RDF for the expression $sin(x + y)$

ing a named reference the function is also available via the URI `<http://example.org/power-2>`.

```
@base <http://example.org/> .
@prefix : <http://numerateweb.org/vocab/math#> .

<square-func> a :Binding ;
  :binder <http://www.openmath.org/cd/fns1#lambda> ;
  :variables (_:x) .
  :body [
    a :Application ;
    :operator <http://www.openmath.org/cd/arith1#power> ;
    :arguments (_:x 2)
  ] .
_:x a :Variable ; :name "x" .
<power-2> a :Reference ; :target <square-func> .
```

Listing 2: Example of a reference to the function $f(x) = x^2$

2.4. Derived objects

To embed non-OpenMath objects into OpenMath objects of type *attribution* or *error derived OpenMath objects* [10, S. 10] can be represented by instances of class `:Foreign`. Its property `:value` has the range `rdf:XMLLiteral` and the property `:encoding` uses an `xsd:string` to specify the content type. This allows to accept simple character strings as well as complete XML documents with nested OpenMath objects.

3. Transformation between OpenMath-XML and OpenMath-RDF

For the transformation from XML to RDF an operator T can be defined. It converts the XML encoding of an OpenMath object O_{XML} to an RDF graph $T(O_{XML})$ containing the equivalent RDF encoding. The rules of the transformation operator T are summarized in Table 1.

The mapping is recursively defined by using the operator T for the top-level element and all of its sub elements. The generated triples by each invocation of T are inserted in the resulting RDF graph. The *main node* in each transformation rule, which is always the subject of the first triple, is the result value of the operator invocation and is used for subsequent transformations.

To accomodate for the differences in the encoding of numbers and URIs between OpenMath-XML and OpenMath-RDF the following helper functions are used to define the operator T:

DEC(HEX) converts a floating point number HEX in hexadecimal encoding into an equivalent decimal representation. This function is necessary because OpenMath-RDF only supports XML-Schema-Datatypes [11] and hence no hexadecimal encodings for floating point numbers.

BASE10(INT) converts an integer INT in decimal or hexadecimal representation to a decimal integer.

RESOLVE(URI) creates an absolute URI according to the rules defined in section 5. "'Reference Resolution"' of the URI specification [12][S. 27 ff.]. This function is necessary because RDF only supports *absolute URIs* as identifiers.

If, for example, the operator T directly creates a Turtle document then the resolution of URIs is not necessary since the Turtle parser resolves URIs automatically against a base URI according to sections 6.3 "'IRI References"' and 7. "'Parsing"' of the Turtle specification [6]. This base URI has to be specified in accordance to the source OpenMath-XML document.

Therefore an OpenMath-XML document at the address `http://example.org/` with the content

```
<OMOBJ><OMR href="named" /></OMOBJ>
```

can be translated into an equivalent Turtle document with the content

```
@base <http://example.org/> .
@prefix : <http://numerateweb.org/vocab/math#> .

[] a :Reference ; :target <named> .
```

The relative URI `<named>` can be kept in the document and by specifying the base URI `@base <http://example.org/>` correctly resolved to an absolute URI by a Turtle parser.

With a few exceptions (numbers, URIs, referenes) the operator T defines an unambiguous mapping between XML and RDF. Therefore an inverse operator T^{-1} for converting RDF

to XML can be easily defined. For handling the exceptions, floating point numbers and integers can either be translated into a decimal or a hexadecimal encoding as OpenMath-XML supports both formats. References to other OpenMath objects can either be directly resolved and copies of the referenced objects in OpenMath-XML format included or `<OMR>` elements can be created with respective relative or absolute URIs.

4. Query OpenMath with SPARQL

OpenMath-RDF allows to use SPARQL [13] as query language to traverse, filter and transform mathematical objects. With SPARQL 1.1 it is also possible to use path expressions for recursive traversals.

As an example the SPARQL query

```
SELECT ?result WHERE {
  ?result (math:arguments|math:symbol|...|rdf:rest)+ ?o .
  {
    ?o <>? <http://www.openmath.org/cd/arith1#sum> .
  } UNION {
    ?o <>? <http://www.openmath.org/cd/arith1#product> .
  } FILTER NOT EXISTS {
    [] math:arguments|math:symbol|...|rdf:rest ?result .
  }
}
```

finds all root expressions that either contain a `sum` or a `product` symbol.

The property path `math:arguments|math:symbol|...|rdf:first` is a shortened version of the path

```
math:arguments|math:symbol|math:operator|math:target|math:variables|
  math:binder|math:body|math:attributeKey|math:attributeValue|
  rdf:rest|rdf:first
```

which ensures that only properties of mathematical objects are traversed by the expression. It would also be possible to just use something like `<>|!<>` if it is acceptable to traverse any edge within the RDF graph.

The property path `<>?` is a trick and expected to always be empty. It is used to avoid the repetition of the long property path `math:arguments|math:symbol|...|rdf:first` for traversing the expression and also may lead to faster execution times if the SPARQL engine is not able to properly optimize the queries.

5. Representation of Content Dictionaries in RDF

OpenMath Content Dictionaries are usually encoded as XML documents. In combination with the RDF encoding introduced in the previous sections Content Dictionaries may also be represented as linked data. Figure 3 depicts an ontology to represent the relevant elements.

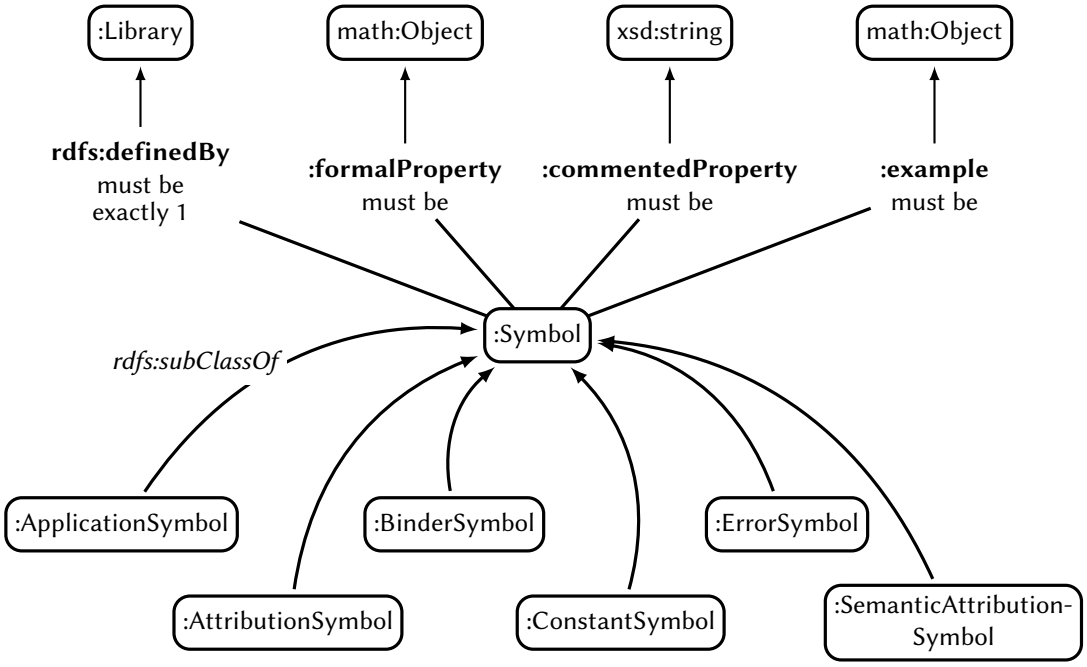


Figure 3: Ontology for meta data of OpenMath Content Dictionaries

The core of the vocabulary are classes for different types of mathematical symbols as defined by the OpenMath standard which are represented by subclasses of `Symbol`. Each symbol is defined (`rdfs:definedBy`) by a Content Dictionary that the ontology models as `Library`. Formal properties (`formalProperty`) of the symbols and usage examples (`example`) refer to mathematical objects as defined by the OpenMath-RDF ontology.

To verify the RDF encoding based on OpenMath-RDF and the meta data ontology 214 Content Dictionaries with 1578 symbols published on the OpenMath web site were converted to an RDF representation[1].

As an example for leveraging the RDF representation of mathematical objects for knowledge management, a catalogue application for Content Dictionaries was developed. This web application is based on the software platform *eniLINK*[2] and allows to browse Content Dictionaries and the contained symbols. Figure 4 shows a screenshot for the symbol `arith1:times`. The mathematical expressions are rendered as Presentation MathML and as a modified version of the POPCORN [14] syntax that is compatible with OpenMath-RDF.

6. Discussion and future work

We have presented RDF encodings for OpenMath objects and Content Dictionaries. These enable the integration of standardized mathematical expressions into RDF-based knowledge

[1]The RDF version of the Content Dictionaries is available at https://github.com/numerateweb/openmath-cd (29.06.2021)

[2]http://platform.enilink.net/ (29.06.2021)

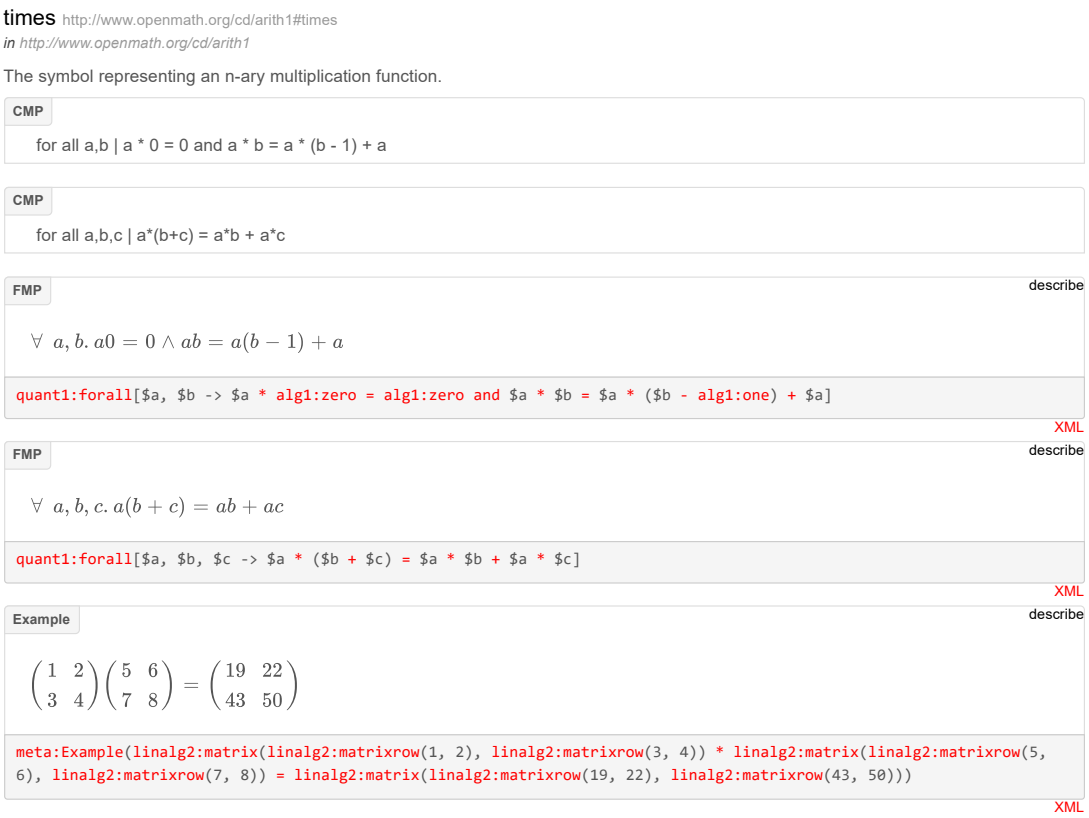


Figure 4: Presentation of symbol `arith1:times` in the web application

graphs. Therefore RDF databases and query engines can be used for storing, querying and transforming OpenMath-based content.

Although the developed vocabularies provide a starting point for integrating OpenMath and the Web of Data some open questions should be addressed.

The ontologies use speaking names for object types instead of acronyms as used by the XML and JSON encodings. This increases the readability of the defined terms and resembles the usual style of ontologies but breaks with the typical terminology of OpenMath.

The representation of literals does currently not support hexadecimal encodings. It should be investigated if hexadecimal representations are necessary and if corresponding RDF datatypes should be defined.

The RDF encoding of Content Dictionaries introduces the term `Library` and also reuses RDF vocabulary like `rdfs:definedBy`. It should be investigated if the encoding should be more closely aligned with the existing XML representation.

References

[1] M. Marchiori, The Mathematical Semantic Web, in: A. Asperti, B. Buchberger, J. H. Davenport (Eds.), Mathematical Knowledge Management, volume 2594, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 216–223. URL: http://www.springerlink.com/content/c087q2550g8k3867/.

[2] G. Schreiber, Y. Raimond, RDF 1.1 Primer, W3C Working Group Note (2014). URL: http://www.w3.org/TR/rdf11-primer/.

[3] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation (2014). URL: http://www.w3.org/TR/rdf11-concepts/.

[4] C. Lange, Enabling Collaboration on Semiformal Mathematical Knowledge by Semantic Web Integration, Ph.D. thesis, Jacobs University Bremen, 2011.

[5] S. Ferré, An RDF Vocabulary for the Representation and Exploration of Expressions with an Illustration on Mathematical Search, 2012. URL: https://hal.inria.fr/hal-00812197.

[6] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, G. Carothers, RDF 1.1 Turtle, W3C Recommendation (2014). URL: http://www.w3.org/TR/turtle/.

[7] E. Muñoz, E. Capón-García, J. M. Laínez-Aguirre, A. Espuña, L. Puigjaner, Using mathematical knowledge management to support integrated decision-making in the enterprise, Computers & Chemical Engineering 66 (2014) 139–150. URL: http://www.sciencedirect.com/science/article/pii/S0098135414000738. doi:10.1016/j.compchemeng.2014.02.026.

[8] K. Wenzel, H. Reinhardt, Mathematical Computations for Linked Data Applications with OpenMath, in: Joint Proceedings of the 24th OpenMath Workshop, the 7th Workshop on Mathematical User Interfaces (MathUI), and the Work in Progress Section of the Conference on Intelligent Computer Mathematics, volume 921, CEUR Workshop Proceedings, Bremen, 2012, pp. 38–48. URL: http://ceur-ws.org/Vol-921/openmath-01.pdf.

[9] M. Horridge, P. F. Patel-Schneider, OWL 2 Web Ontology Language Manchester Syntax, W3C Working Group Note 5. September 2012 (2012). URL: https://www.w3.org/2007/OWL/draft/owl2-manchester-syntax/.

[10] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, M. Kohlhase, J. H. D. Davenport, P. D. F. Ion, T. Wiesing, The OpenMath standard, Technical Report, The OpenMath Society, 2019. URL: https://openmath.org/standard/om20-2019-07-01/omstd20.pdf, version 2.0 revision 2.

[11] D. Peterson, S. Gao, A. Malhotra, C. M. Sperberg-McQueen, H. S. Thompson, W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, W3C Recommendation (2012). URL: http://www.w3.org/TR/xmlschema11-2/.

[12] T. Berners-Lee, R. Fielding, L. Masinter, RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, 2005. URL: http://www.ietf.org/rfc/rfc3986.

[13] S. Harris, A. Seaborne, E. Prud'hommeaux, SPARQL 1.1 Query Language, W3C Recommendation (2013). URL: http://www.w3.org/TR/sparql11-query/.

[14] P. Horn, D. Roozemond, OpenMath in SCIEnce: SCSCP and POPCORN, in: Intelligent Computer Mathematics, Springer, 2009, pp. 474–479.

Table 1
Translation between OpenMath-XML and OpenMath-RDF

OpenMath-XML – O_{XML}	**OpenMath-RDF** – $T(O_{XML})$
Basic objects	
`<OMF dec="DEC" />`	`_:l a :Literal ; :value "DEC"^^xsd:double .`
`<OMF hex="HEX" />`	`_:l a :Literal ;` `  :value "DEC(HEX)"^^xsd:double .`
`<OMI>INT</OMI>`	`_:l a :Literal ;` `  :value "BASE10(INT)"^^xsd:integer .`
`<OMSTR>STRING</OMSTR>`	`_:l a :Literal ; :value "STRING" .`
`<OMB>BYTES</OMB>`	`_:l a :Literal ;` `  :value "BYTES"^^xsd:base64Binary`
`<OMV name="NAME" />`	`_:o a :Variable ; :name "NAME" .`
`<OMS cd="CD" name="NAME" />`	`<CDBASE/CD#NAME> a :Symbol .`
Compound objects	
`<OMA>OP A1 ... An</OMA>`	`_:c a :Application ; :operator T(OP) ;` `  :arguments (T(A1) ... T(An)) .`

`<OMBIND>`
`  B`
`  <OMBVAR>V1 ... Vn</OMBVAR>`
`  C`
`</OMBIND>`
	`_:c a :Binding ;`
	`  :binder T(B) ; :body T(C) ;`
	`  :variables (T(V1) ... T(Vn)) .`

`<OMATTR>`
`  <OMATP>`
`  S1 A1 ... Sn An`
`  </OMATP>`
`  X`
`</OMATTR>`
	`_:c a :Attribution ;`
	`  :target T(X) ; :arguments (`
	`  [ :attributeKey T(S1) ;`
	`  :attributeValue T(A1) ]`
	`  ...`
	`  [ :attributeKey T(Sn) ;`
	`  :attributeValue T(An) ] ) .`

`<OME>S A1 ... An</OME>`	`_:c a :Error ; :symbol T(S) ;` `  :arguments (T(A1) ... T(An)) .`
Named objects & references	
`<... id="URI" />`	`<RESOLVE(URI)> a ... .`
`<OMR href="URI" />`	`_:o a :Reference ; :target <RESOLVE(URI)> .`
Derived objects	

`<OMFOREIGN encoding="ENC">`
`  BODY`
`</OMFOREIGN>`
	`_:o a :Foreign ;`
	`  :encoding "ENC" ;`
	`  :value "BODY"^^rdf:XMLLiteral .`