

## SBML Level 3 Package Specification

# Math-SBO

People      A N Other

Version 1 Release 0.1 (Draft)

17th May 2014

The latest release, past releases, and other materials related to this specification are available at

[TBC](#)

*This* release of the specification is available at

[TBC](#)



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Package dependencies . . . . .	3
<b>2</b>	<b>Package syntax and semantics</b>	<b>4</b>
2.1	Extending existing math elements . . . . .	4
2.2	Namespace URI and other declarations necessary for using this package . . . . .	4
2.3	Additional MathML Constructs . . . . .	5
2.3.1	Type attribute on the ci element . . . . .	5
2.3.2	The Selector element . . . . .	5
2.3.3	The Lowlimit and Uplimit elements . . . . .	5
2.3.4	The Sum and Product elements . . . . .	5
<b>A</b>	<b>Validation of SBML documents</b>	<b>7</b>
A.1	Validation and consistency rules . . . . .	7
	<b>Acknowledgments</b>	<b>8</b>
	<b>References</b>	<b>9</b>

---

# 1 Introduction

---

## 1.1 Motivation

The need to extend the subset of MathML supported within SBML is becoming increasingly apparent. However to facilitate the adoption of sets of math constructs these are being introduced as a series of small packages each focussing on one area of mathematical constructs.

This package (math-sbo) adds those MathML constructs that are present in the MathML subset supported by the Systems Biology Ontology (SBO) (?).

## 1.2 Package dependencies

The Math-SBO package has no dependencies on other SBML Level 3 packages. (If you find incompatibilities with other packages, please contact the Package Working Group. Contact information is shown on the front page of this document.)

## 2 Package syntax and semantics

In this section, we define the syntax and semantics of the Math SBO package for SBML Level 3. We expound on the various data types and constructs defined in this package.

### 2.1 Extending existing math elements


Any of the SBML Level 3 Math packages add constructs from MathML 2.0 (?) to the subset of MathML supported by SBML. These constructs are used within the existing math elements that occur within other SBML constructs and are placed within the mathml namespace in the same manner as has always been adopted by SBML. Thus the new selector construct added by this package would be used in SBML as shown in the example.

```
<assignmentRule variable="m">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply>
      <selector/>
      <ci> f </ci>
    </apply>
  </math>
</assignmentRule>
```

### 2.2 Namespace URI and other declarations necessary for using this package

Every SBML Level 3 package is identified uniquely by an XML namespace URI. For an SBML document to be able to use a given SBML Level 3 package, it must declare the use of that package by referencing its URI. The following is the namespace URI for this version of the Math SBO package for SBML Level 3:

`"http://www.sbml.org/sbml/level3/math/sbo/version1"`

 **Note** I left out the SBML version number on the grounds that this should work for all L3 versions. I debated whether to actually even put a version number for the package - since I cannot image it actually changing.

In addition, SBML documents using a given package must indicate whether understanding the package is required for complete mathematical interpretation of a model, or whether the package is optional. This is done using the attribute **required** on the **<sbml>** element in the SBML document. For any of the SBML Level 3 Math packages, the value of this attribute must be set to **"true"**.

The following fragment illustrates the beginning of a typical SBML model using SBML Level 3 Version 1 and this version of the Math SBO package:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:math-sbo="http://www.sbml.org/sbml/level3/math/sbo/version1" math-sbo:required="true">
```

## 2.3 Additional MathML Constructs

The following are the additional math constructs introduced by this package with explanations of their usage. It should be noted that the interpretation of some constructs may differ depending on other Level 3 packages that may or may not be present. Developers should consider which of these packages are supported when developing support for the math constructs defined here.

### 2.3.1 Type attribute on the ci element

An additional attribute **type** is permitted on a **ci** element. This is optional and of **string** with a single permitted value 'vector'.

 Should we actually create a type for this - I used what the sbo-mathml schema used

### 2.3.2 The Selector element

MathML defines **selector** as the operator used to extract sub-objects from vectors. The syntax involves two arguments, the first representing a vector and the second representing an integer value that is the index into the vector supplied.

The example snippet illustrates the MathML for extracting the object at index 2 from the vector f i.e.

$$f[2]$$

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <selector/>
    <ci type="vector"> f </ci>
    <cn type="integer"> 2 </cn>
  </apply>
</math>
```

It should be noted that the SBML Level 3 Core considers all identifiers to refer to single valued constructs. The arrays package (?) allows some constructs within SBML to be given dimensions and thus treated as vectors.

 Need some more explanation of how the above might be used with just core - NLN ???

 In the situation where the identifier refers to an object of single value that value should be returned. do we want to specify this ??

### 2.3.3 The Lowlimit and Uplimit elements

MathML defines **lowlimit** and **uplimit** as a means of applying upper and lower limits to an interval. Here we restrict the syntax to allow only one child element that can be either a **ci** element referring to an SBML object that gives the value for the limit or a **cn** element that explicitly defines the value. Again this is what the sbo mathml schema does - do we apply this restriction



### 2.3.4 The Sum and Product elements

MathML defines **sum** as the summation operator and **product** as the product operator. Upper and lower limits for the index can be specified using **uplimit** and **lowlimit**. 'notice MathML spec adds: More general domains for the indices can be specified using a condition involving the bound variables. - should we The index for the summation is specified by a **bvar** element. the sbo mathml schema does not actually allow bvar on sum but only on product



The example snippet illustrates the MathML for summing the contents of vector f from index 1 to 3 i.e.

$$f[1] + f[2] + f[3]$$

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
```

```

<apply>
  <sum/>
  <bvar><ci> x </ci></bvar>
  <lowlimit><cn> 1 </cn></lowlimit>
  <uplimit><cn> 3 </cn></uplimit>
  <apply>
    <ci type="vector"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
</math>

```



The mathML spec actually only gives examples of using sum and product with a function so I'm not entirely sure my MathML example is correct. Do we need the apply and indeed the bvar - or should in fact the apply have a selector following it ?

The **sum** and **product** operators can be used with core constructs as a replacement for **plus** or **times** as in the following snippet.

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <product/>
    <ci> g </ci>
    <cn> 4 </cn>
  </apply>
</math>

```



would there be a way of using them with distrib ???

## A Validation of SBML documents

### A.1 Validation and consistency rules

This section summarizes all the conditions that must (or in some cases, at least *should*) be true of an SBML Level 3 Version 1 model that uses the Qualitative Models package. We use the same conventions as are used in the SBML Level 3 Version 1 Core specification document. In particular, there are different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition *must* be true, or a rule states that it *should* be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules. To help highlight these differences, we use the following three symbols next to the rule numbers:

- ☑ A checked box indicates a *requirement* for SBML conformance. If a model does not follow this rule, it does not conform to the Qualitative Models specification. (Mnemonic intention behind the choice of symbol: “This must be checked.”)
- ▲ A triangle indicates a *recommendation* for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the Qualitative Models specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: “This is a cause for warning.”)
- ★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not strictly considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: “You’re a star if you heed this.”)

The validation rules listed in the following subsections are all stated or implied in the rest of this specification document. They are enumerated here for convenience. Unless explicitly stated, all validation rules concern objects and attributes specifically defined in the Qualitative Models package.

- 🗉 For convenience and brevity, we use the shorthand “**qual:x**” to stand for an attribute or element name **x** in the namespace for the Qualitative Models package, using the namespace prefix **qual**. In reality, the prefix string may be different from the literal “**qual**” used here (and indeed, it can be any valid XML namespace prefix that the modeler or software chooses). We use “**qual:x**” because it is shorter than to write a full explanation everywhere we refer to an attribute or element in the Qualitative Models package namespace.

#### General rules about this package

- sbo-10101** ☑ To conform to the Math SBO package specification for SBML Level 3, an SBML document must declare the use of the following XML Namespace:  
“<http://www.sbml.org/sbml/level3/math/sbo/version1>”. (References: SBML Level 3 Package Specification for Math SBO, Version 1, [Section 2.2 on page 4](#).)

#### General rules for MathML content

---

# Acknowledgments

---



---

## References

---

Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at <http://www.w3.org/TR/xmlschema-2/>.

Eriksson, H.-E. and Penker, M. (1998). *UML Toolkit*. John Wiley & Sons, New York.

Fallside, D. C. (2000). XML Schema part 0: Primer (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at <http://www.w3.org/TR/xmlschema-0/>.

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., Schaff, J. C., Smith, L. P., and Wilkinson, D. J. (2010). The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. Available via the World Wide Web at <http://sbml.org/Documents/Specifications>.

Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures (W3C candidate recommendation 24 October 2000). Available online via the World Wide Web at the address <http://www.w3.org/TR/xmlschema-1/>.