Proposal for multiple small additional math packages for SBML Level 3

Sarah Keating

26th October 2014

Edited 8th April 2015

Motivation

Currently SBML restricts the subset of MathML that is deemed 'valid' for use within an SBML model. It was considered that L3 packages that needed additional math would add this as required. However, this limits the use of the additional math to the use of the package - which in some cases may not be necessary. For example the arrays package has suggested including the mathematical operation 'mean' - but it would be perfectly feasible to use 'mean' without using arrays.

Discussions of additional math constructs that should be added to core SBML inevitably introduces difference of opinion as there are varying views on what is considered useful and/or necessary.

It has also been suggested that we relax the restriction and just allow all of MathML to be used within SBML. However, learning from the experience of CellML, this may produce a situation where many software packages do not support all mathematical operations. This can lead to reduced interoperability with some models only being simulatable by one software package.

This proposal attempts to provide a medium ground where we facilitate modelers in their need to use mathematical constructs whilst maintaining a position where software supports this and we can continue to have the level of interoperability that SBML has currently achieved.

Proposal

This proposal suggests that we produce a number of small math only packages that group together math constructs that would be added by that package e.g. statistical functions, array functions, linear algebra functions (see Appendix A for some suggested lists of groupings). Each package would have its own sbml namespace and use the required attribute to indicate that it has been used in a fashion consistent with the current use of packages.

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"

  stats:xmlns="http://www.sbml.org/sbml/level3/math/statistics" stats:required="true">
```

However the package namespace would only be used to indicate the use of additional constructs. The MathML constructs themselves would remain in the MathML namespace in accordance with the current use of MathML within SBML.

Advantages

Any math construct is only defined once and can be used by any package that needs it by declaring a dependency on the particular math package. This reduces the possibility of ambiguity if multiple L3 packages add the same additional math constructs.

Software can choose which math packages to support which reduces the onus on developers and the package namespace and required attribute mechanism enables software to easily determine and report to users whether the constructs are supported.

The additional math would be usable with SBML L3V1 core without the need to support another package or version of SBML.

Disadvantages

Discussion of this idea has raised the issue of implementation support. In the case of some mathematically constructs the interpretation may be ambiguous depending on the context. For example, the recent discussion between developers as to the meaning of a relational operator with three or more arguments showed that individual interpretation of the same thing may differ. Since MathML itself was developed with presentation in mind, rather than strict mathematical interpretation, it is difficult to rely on the MathML specifications for unambiguous interpretation. (I refer people to the discussion at HARMONY 2014 about the MathML definition of ‘mode’.)

Tackling this issue

In order to overcome the possibility of ambiguous interpretation each specification would outline the accepted “SBML” meaning of a construct, in terms of any of the Level 3 packages under development. An example of the section on ‘mean’ is given in the Appendix B to this document. Note the mathematical interpretation given here is merely an attempt to illustrate the types of information to be given.

Rationale for this solution

This approach requires a large amount of effort, compared to that involved in just listing a set of mathematical operators that can be used by each of these mini-math packages. However, the history of SBML development has shown that if the interpretation of a construct has ambiguity then either people will not attempt to implement support for it at all or interpretation may vary and different implementations will produce different results. Neither of these situations is desirable.

It could be argued that by defining a precise SBML interpretation for each construct we are imposing further restriction on the modeller rather than providing greater freedom. However, a counter argument to this is that by clarifying the interpretation in this way we are providing the modeller with a truly exchangeable standard.

Appendix A: Possible grouping of MathML functions into packages

(Note: These lists contain all the content elements of MathML 2.0 that are not yet supported in SBML. The categorisations and divisions are NOT definitive – merely a suggested starting point. Indeed there would still need to be decisions on whether a given category was useful at all etc.)

stats	**arrays**	**linear algebra**	**sum/product**
mean	vector	matrix	sum
variance	selector	matrixrow	product
mode		inverse	lowlimit
sdev		transpose	uplimit
median		determinant	tendsto
moment		vectorproduct	
momentabout		scalarproduct	
		outerproduct	

content	**logic**	**arithmetic**	**sets**
interval	condition	quotient	emptyset
	forall	max	set
	exists	min	list
	equivalent	rem	union
		lcm	intersect
		gcd	in
		implies	notin
		factorof	subset
		approx	notsubset
			prsubset
			notprsubset
			setdiff
			card
			cartesianproduct

complex	**function composition**	**calculus**	**others ?**
conjugate	compose	int	ident
arg	domain	diff	naturalnumbers
real	codomain	partialdiff	primes
imaginary	domainofapplication	curl	eulergamma
complexes	image	divergence	
imaginaryi		laplacian	
		grad	

Appendix B: Proposed section for the <mean> function

(NOTE: This example is meant to illustrate the idea – not actually define the use of <mean>.)

MathML specification

(Extract from the MathML-2.0 specification http://www.w3.org/TR/2002/WD-MathML2-20021219/appendixc.html#cedef.mean)

Description

The mean value of a set of data, or of a random variable. See CRC Standard Mathematical Tables and Formulae, editor: Dan Zwillinger, CRC Press Inc., 1996, section 7.7.1

See also **4.4.9.1 Mean (mean)**.

Classification

function

MMLattribute

Name	Value	Default
definitionURL	URI identifying the definition	APPENDIX_C
encoding	CDATA	MathML

Signature

(random_variable) -> scalar

(scalar+) -> scalar

SBML interpretation

SBML interprets <mean> as an n-ary function.

[State whether the attributes ‘definitionURL’ and ‘encoding’ are allowed and if so what meaning and/or values they have.]

Applied to core:

When used with SBML L3 core the <mean> function would expect to have zero or more arguments that represent scalars.

e.g.

```
<math>
        <apply>
                <mean/>
                <ci> s </ci>
                <cn> 2.4 </cn>
                <apply><plus/><cn>2</cn>>cn>3</cn>
        </apply>
</math>
```

[Define the result of mean applied to 0 arguments.]

When applied to one argument it returns that argument.

When applied to two or more arguments it returns the arithmetic mean of the values.

Applied to arrays:

[Define what arguments would be expected e.g. When used with SBML L3 arrays the <mean> function would expect to have [one argument ?] which would be a ci element pointing to the id of an object specified as an array.]

e.g.

```
<parameter id=”p” constant=”false”>
        <arrays:listOfDimensions>
                <arrays:dimension size=”9” arrayDimension=”0”/>
        </arrays:listOfDimensions>
</parameter>
…
<math>
        <apply>
                <mean/>
                <ci> p </ci>
        </apply>
</math>
```

It returns the arithmetic mean of all the values represented by the array.

[Decide whether it applies to two or more arrays arguments and if so define the result of mean applied to 2+ array arguments.]

Applied to distrib:

[It would have to be decided if 'mean' with a single argument to a distrib-extended function definition meant that the mean of the function itself was being described, or if the mean of the draw *from* the function was being described.]

[Decide whether it applies to two or more distrib arguments and if so, define the result of mean applied to 2+ distrib arguments.]

[Decide what if mean can be applied to an object with a distrib:uncertainty element and if so what the result is…]

So basically the idea would be that we make a clear definition of what a construct means when applied to scalars and then to any other SBML constructs that represent something that is not a scalar.