# Required Elements

At the heart of many SBML models is mathematics. When present, this mathematics is considered to be of principle importance to understanding the model. Many SBML Level 3 packages extend the mathematical concepts present in SBML Level 3 core, and when they do, a model that includes these concepts must add a ‘required=true’ flag on the definition of that namespace.

However, for tools that wish to manipulate and augment SBML models with extended math which they do not understand, a blanket ‘required=true’ flag on the namespace is not granular enough. What particular elements have had their mathematics changed? What bits are ‘safe’ to manipulate freely, and which must be treated with care?

This very small package allows modelers to declare specifically which elements of the model have had their math changed, by whom, and if a core alternative is present, by the addition of two optional attributes to SBase.

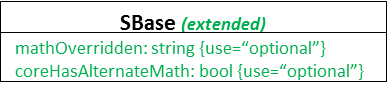


Figure : Definition of the SBase extension. Two attributes are added: ‘mathOverridden’, a string, and ‘coreHasAlternateMath’, a Boolean. Both are optional, and are only legal for SBase derived classes with a ‘Math’ subelement.

The first attribute is ‘mathOverriden’ and is set to the namespace of a package that redefines that element’s math. This must be the element whose math is directly affected, not any element whose math is indirectly affected. For example, if a functionDefinition overrides the math of ‘random(mean, variance)’, and an InitialAssignment uses that function to set the initial value of a species, only the functionDefinition itself should have the ‘mathOverridden’ attribute set—the InitialAssignment and the species itself are only indirectly affected, and may not set the ‘mathOverridden’ element.

The second attribute is ‘coreHasAlternateMath’, and is set to be true or false, depending on whether an interpreter that only understood core would have a workable and complete, if different, version of the math for that element. ‘Complete’ is an objective requirement for setting this attribute to ‘true’—the provided core elements must not leave any of the math undefined. ‘Workable’ is a judgment call on the part of the modeler—if they feel the alternate version makes sense in an alternate context, they may set that attribute to true, and if they feel the resulting element makes no sense: even if it is technically ‘complete’, they may set that attribute to ‘false’.

As an example, we present a few functionDefinitions whose math has been changed by the (as-yet-hypothetical) ‘distrib’ package:

<listOfFunctionDefinitions>

<functionDefinition id="unitGaussian" req:mathOverridden="distrib" req:coreHasAlternateMath=true>

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

xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">

<lambda>

<cn> 0.5 </cn>

</lambda>

</math>

<distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"

function="gaussian" mean="0.5" variance="0.25"

</distrib:call>

</functionDefinition>

</listOfFunctionDefinitions>

In this case, the function ‘unitGaussian()’ has both a core definition (‘return 0.5’) and a package-defined version (‘select a random number from a gaussian distribution with mean 0.5 and variance 0.25’). The modeler has elected to claim that the core definition’s version of the function is adequate for the model in which it is being used, although different results are to be expected.

Here is a similar function where the modeler has decided that the core version of the math will not make an adequate replacement:

<listOfFunctionDefinitions>

<functionDefinition id="gaussian" req:mathOverridden="distrib" req:coreHasAlternateMath=false>

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

xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">

<lambda>

<bvar> <ci> mean </ci> </bvar>

<bvar> <ci> variance </ci> </bvar>

<cn> 0 </cn>

</lambda>

</math>

<distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"

function="gaussian" mean="mean" variance="variance"

</distrib:call>

</functionDefinition>

</listOfFunctionDefinitions>

Finally, the following version \*must\* be maked 'coreHasAlternateMath=false' because the lambda function is incomplete:

<listOfFunctionDefinitions>

<functionDefinition id="gaussian" req:mathOverridden="distrib" req:coreHasAlternateMath=false>

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

xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">

<lambda>

<bvar> <ci> mean </ci> </bvar>

<bvar> <ci> variance </ci> </bvar>

</lambda>

</math>

<distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"

function="gaussian" mean="mean" variance="variance"

</distrib:call>

</functionDefinition>

</listOfFunctionDefinitions>

Note: These attributes are added to SBase instead of to those elements which have math in them specifically (parameters, species, triggers, etc.) because this allows packages to define new elements with math which may, in turn be overridden by other packages.

The ‘Required Elements’ package is not, itself, required (that is, models that use it must set ‘required=false’ for its namespace when using it) because it, in and of itself, does not change any math.

However, it is intended to be used by other required packages, which may in turn make the use of one or both attributes from this package required in certain contexts defined in their own specifications. For example, the ‘distrib’ package could require that modelers set the attribute ‘mathOverridden=”distrib”’ for any functionDefinition whose math was defined by that package. This would provide a clean way of ensuring that only one package overrode the math for any single SBML element: if both packages required the 'mathOverridden' attribute to be set to their own namespace, it would be impossible to doubly-override a single element without failing to comply with at least one package specification.

It is anticipated that future versions of SBML Level 3 may incorporate the ‘Required Elements’ package into the core specification itself, should it prove generally useful.