1 CELLML2SBML: CONVERSION OF CELLML INTO SBML

Running head: Conversion of CellML into SBML

Authors:

Maria J. Schilstra*
Biological and Neural Computation Group, STRI
University of Hertfordshire
Hatfield, AL10 9AB
UK

Lu Li
Computational Neurobiology
EMBL-EBI
Wellcome-Trust Genome Campus
Hinxton, Cambridge, CB10 1SD
UK

Joanne Matthews
Biological and Neural Computation Group, STRI
University of Hertfordshire
Hatfield, AL10 9AB
UK

Andrew Finney
Physiomics PLC
Magdalen Centre, Oxford Science Park
Oxford, OX4 4GA
UK

Michael Hucka
Control and Dynamical Systems
California Institute of Technology
Pasadena, CA 91125
USA

Nicolas Le Novère
Computational Neurobiology
EMBL-EBI
Wellcome-Trust Genome Campus
Hinxton, Cambridge, CB10 1SD
UK

* To whom correspondence should be addressed

ABSTRACT

Summary: CellML and SBML are XML-based languages for storage and exchange of molecular biological and physiological reaction models. They use very similar subsets of MathML to specify the mathematical aspects of the models. CellML2SBML is implemented as a suite of XSLT stylesheets that, when applied consecutively, convert models expressed in CellML into SBML without significant loss of information. The converter is based on the most recent stable versions of the languages (CellML version 1.1; SBML Level 2 Version 1), and the XSLT used in the stylesheets adheres to the XSLT version 1.0 specification. Of all 306 models in the CellML repository in April 2005, CellML2SBML converted 91% automatically into SBML. Minor manual changes to the unit definitions in the originals raised the percentage of successful conversions to 96%.

Availability: http://sbml.org/software/cellml2sbml/

Contact: m.j.1.schilstra@herts.ac.uk

Supplementary information: Instructions for use and further documentation available on http://sbml.org/software/cellml2sbml/

2 INTRODUCTION

The Systems Biology Markup Language (SBML, Finney and Hucka, 2003, Hucka, et al. 2003) and the Cell Markup Language (CellML, Cuellar, et al., 2003) are XML-based markup languages designed to capture the structure and content of models of molecular biological and physiological systems. They are used for model exchange between software tools, and for storage in a standard format. The scope of CellML ("to describe the mathematics behind any biological model") is somewhat wider than that of SBML ("to represent models of biochemical reaction networks"), but, fortuitously, both use almost identical mathematical expressions, and specify these using an almost identical subset of MathML. Figure 1 shows the relevant CellML and SBML elements, and gives a rough indication of the mapping. In this figure the CellML `import` element has not been taken into account, whereas the SBML elements `functionDefinition` and `event` are irrelevant in this context, and the containment structures (discussed below) are conceptually so different that they had to be ignored.

2.1 SBML Model Extraction

Identifiers: CellML uses a `name` attribute to uniquely identify model components, whereas SBML uses an `id` attribute for the same purpose (the optional `name` attribute in SBML is used as a label); `cellml:name` and `sbml:id` attributes are subject to the same limitations. Therefore, global CellML `name` attribute values are simply retained as SBML `id` values; local `name` values (those of `variable` elements and of `units` defined inside a `component`) are converted into a combination of the local `name` and the `name` of the parent `component`. In that case, the converter retains the original CellML `name` under the SBML `name` attribute.

Units: The CellML `units` and SBML `unitDefinition` structures, which are used for the definition of composite units, both contain a list of `unit` elements. The `unit` elements in both languages have essentially the same attributes, and use the same set of basic (SI) units, so that conversion is straightforward. However, where CellML permits nested use of `units` elements, SBML (Level 2 Version 1) requires its units to be defined in terms of the basic SI units. Therefore, the CellML2SBML converter expands the `units`, and combines the resulting base units as appropriate in a new SBML `unitDefinition`. A special `unitDefinition` element is created to specify the time base for ODEs (ordinary differential equations; in `rateRule` and `kineticLaw`) if the time base used in the CellML model differs from the SBML default unit of time. Furthermore, the `cellml:units` associated with `variable` elements that convert to `species` (see below) are decomposed into units of substance and units of spatial size, and corresponding `unitDefinition` elements are added to the list (if necessary). These can then be referred to in the `substanceUnits` and `spatialSizeUnits` attribute fields of `species`.

Compartments: SBML and CellML can both express physical containment: CellML in its nested `group` structure, and SBML through its compartment element and `outside` attribute. However, SBML requires explicit definition of compartment size, and computation of concentration changes associated with inter-compartmental transport is left to the interpreter. CellML lacks fields to specifically delimit compartment dimensions, and any transformation rules must be explicitly defined. Therefore, automatic conversion between a CellML `group` and SBML compartment is impossible, and the converter simply generates a single compartment, with which it then associates all species (see below).

Species and parameters: All `variable` instances that, through a CellML `variable_ref` structure, assume the role of "reactant", "product", "modifier", "catalyst", "inhibitor", or "activator" in a `cellml:reaction`, are converted into `sbml:species` instances; all other `variable` instances are converted, in the first instance (but see the section on model reduction), to `sbml:parameter`.

Reactions and rules: A CellML `reaction` (contained in a `component`) is conceptually almost equivalent to an SBML `reaction`. A CellML `variable_ref` element whose `role` is “reactant” or “product” is converted to a `speciesReference` in an SBML `reaction`; one with a “modifier”, “catalyst”, “inhibitor”, or “activator” `role` becomes a `modifierSpeciesReference`. The `variable_ref` element whose `role` is “rate” is converted into a `kineticLaw` element, whereby the right-hand side of the MathML `math` block inside the `variable_ref` is copied into the `math` element of the new `kineticLaw`. The kineticLaw math must be expressed in units of substance over time. If the rate in the original CellML is in units of concentration over time, a multiplier to the kinetic law expression is generated, and a local parameter of value 1, and with the units of volume (or area, or length, where appropriate) is added to the `reaction` element. The `variable_ref` elements with `role` “delta_variable” are dealt with in the model reduction phase (below).

The `math` blocks that are direct children of `component` instances (i.e., are not contained in a `role` element within a CellML `reaction`) are converted to the appropriate `rule`: an `assignmentRule` when the left-hand side of the equation is a single token (MathML `ci`) that refers to a variable, and a `rateRule` when the left-hand side encodes a time-based ODE. Furthermore, all CellML `connection` elements, which effectively specify equalities, are, in the first instance, converted to `assignmentRule` instances, thereby making the equalities explicit.

2.2 SBML Model Reduction and Adjustment

After a first-pass conversion as described above, the resulting model is likely to contain a high degree of redundancy, and is, at this stage, not necessarily syntactically valid SBML. In the next stage of the CellML2SBML conversion process, redundancy is reduced by tracking the chains of input-output connections to their source, substituting the `id` of the elements in the chain and their references with the `id` of the source, removing all `assignmentRule` instances that, as a result, specify an identity, and removing multiple copies of `parameter` or `species` instances. If, after reduction, the original CellML identifier that was retained in the SBML `name` is found to be unique across the model, CellML2SBML replaces the current composite `id` with the value of the `name` attribute.

To avoid potential conflict through over-specification, SBML, unlike CellML, leaves construction of ODEs from the rate equations in its `kineticLaw` constructs to interpreters. Therefore, the converter removes from the final model each `parameter` that functions solely as a “delta_variable” in a CellML `reaction`, as well as any ODE whose right-hand side is a simple sum of `parameter` instances that are referred to as “delta_variable”.

The SBML specification stipulates that `assignmentRule` instances in the `listOfRules` be listed in topological order (i.e., if a parameter or species appears as an independent variable in one of the equations, it can only function as a dependent variable in equations that are listed later in the sequence). Therefore, a topological sort on the remaining `assignmentRule` instances is carried out as a final step. Circular references are eliminated by a process of replacing one member of the cycle by its equivalent `algebraicRule` (i.e., transformation of $y = f(x)$ into $0 = f(x) – y$) and re-sorting, until all circular references have disappeared.

3 RESULTS

Out of the 306 models in the CellML repository on 5 April 2005, CellML2SBML was able to convert 279 (91%) of the models automatically into valid SBML, and 14 more (5%) could be converted after the 14 models had their newly defined base units (legitimate in CellML, not in SBML) replaced by their equivalent SI units. The validity of the SBML document was assessed by the libSBML-based validator that is provided on-line on http://sbml.org/tools/htdocs/sbmltools.php. Eight models contained ODEs that combined delta-variables with parameters that were not specified as delta-variables, a complication that in many cases may be solved by limited manual modification of the CellML model, such as addition of source or sink components to reactions. The remaining six models had certain aspects, such as partial differential equations, that are incompatible with SBML Level 2 Version 1.

ACKNOWLEDGMENTS

The authors would like to thank Poul Nielsen and Autumn Cuellar of the CellML team, and the SBML Forum, for fruitful discussions.

REFERENCES

Cuellar, A.A., Lloyd, C.M., Nielsen, P.F., Bullivant, D.P., Nickerson, D.P. and Hunter, P.J. (2003) An Overview of CellML 1.1, a Biological Model Description Language., SIMULATION: Transactions of The Society for Modeling and Simulation International., 79, 740–747.

Finney, A., and Hucka, M. (2003) Systems Biology Markup Language: Level 2 and Beyond, *Biochemical Society Transactions* 31:1472–1483.

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H. and the rest of the SBML Forum (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, Bioinformatics, 19, 524–531.

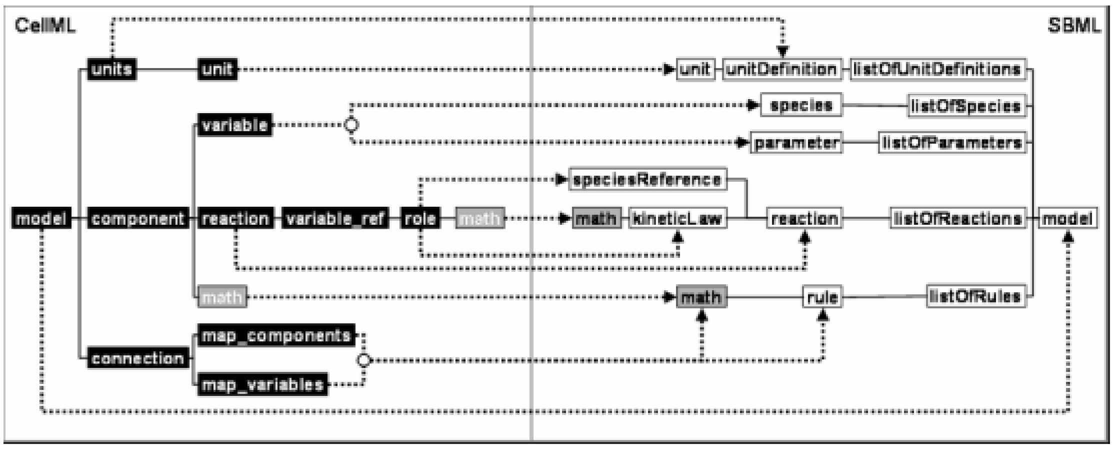

Fig. 1 Class structure of CellML (black boxes) and SBML (white boxes), and their MathML elements (gray boxes). The CellML `import` and `group`, and the SBML `compartment` and `event` elements have been omitted (see text). Solid lines indicate encapsulation (a CellML element is encapsulated by the elements on its left; the SBML elements by the one on its right). Dashed arrows indicate how the CellML elements translate into their corresponding SBML elements.