Meeting Minutes 22 April 2001
Review of Conformance Levels in CellML

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Melanie Nelson (Physiome Sciences Inc.)
Poul Nielsen (Bioengineering Institute, University of Auckland)

1 Introduction

Recent progress on the units and mathematics parts of the CellML specification has caused some questions to be asked about the current system of conformance levels in CellML. This document presents some background about the issue, and some options. It will hopefully stimulate discussion. **Addendums:** On 23 April 2001, Poul and David officially ratified the recommendations of this document. On 24 April 2001, it was ratified by Melanie Nelson at Physiome, who added a few items to the terminology list.

2 The Current Two-Level System

Section 1.2.2 of the 2001-03-02 draft of the CellML specification proposes a two-level system of CellML conformance for both documents and processing software. The intent of the two-level system was to separate out some features of CellML that were considered too hard to require everybody to implement, and yet define them to guarantee interoperability at the "extra for experts" level. Furthermore, processors are only required to implement rules that are deemed "appropriate" to be able to claim CellML compliance. The relevant technical rules are quoted below:

> **CellML conformance level one** A CellML Document is conformant to level one of the CellML specification if it complies with all level one rules for documents in the CellML specification.
>
> A CellML processor is conformant to level one of the CellML specification if it can validate CellML documents against all level one rules for documents in the CellML specification and it follows all *appropriate* level one rules for processor behaviour in the CellML specification when interpreting CellML documents. The *appropriate* rules are those that relate to the intended use of the software (i.e., software that only renders the model need not address the scope of units definitions).
>
> **CellML conformance level two** A CellML document is conformant to level two of the CellML specification if it complies with all level one and level two rules for documents in the CellML specification.
>
> A CellML processor is conformant to level two of the CellML specification if it can validate CellML documents against all level one and level two rules for documents in the CellML specification and it follows all *appropriate* level one and level two rules for processor behaviour in the CellML specification when interpreting CellML documents. The *appropriate* rules are those that relate to the intended use of the software (i.e., software that only renders the model need not address the consistency of mathematics).

3 The Current Level Two Features

The current areas of CellML which involve two conformance levels because we can not realistically expect everyone to implement them are:

- **Subsetting of MathML** In the interests of making the implementation of MathML processing as simple as possible, and to improve interoperability, the CellML specification will define a subset of the MathML content markup elements which CellML processing software will be required to correctly interpret. CellML Level 1 conformant software is required to interpret all elements in the subset, whereas CellML Level 2 conformant software is required to interpret all MathML content elements. (Note that I don't expect anyone to ever do this, but by putting this in writing, we indicate the direction that we're moving in.)
- **Function Calls** The CellML specification will not define any mechanism for calling functions from within blocks of MathML. However, in order to ensure future interoperability, we define some Level 2 rules regarding the use of the `<mathml:csymbol>` element, which is not in the CellML subset of MathML elements.
- **Conversion Between Units Definitions** Level 1 conformant software will not be required to be able to convert, for instance, inches to centimetres. (It may, however, be able to detect units references that are not equivalent, and alert the user.) If software does choose to convert units definitions, Level 2 conformant software must achieve the same results as if an algorithm that is defined in the specification were applied. This guarantees that software achieves the same results no matter how units are handled internally.
- **Equation Dimension Checking** Level 1 conformant software is not required to check that the dimensions of terms in an equation are self-consistent. Level 2 conformant software may choose to do this, and if so, must achieve the same results as if an algorithm that is defined in the specification were applied. This guarantees that all software will accept and reject (or signal a possible error) the same equations.

4 A Better Method?

The concept of levels was introduced primarily because we want to define some behaviour that we don't expect everyone to implement, and at the same time we want people to be able to claim "complete CellML compliance". We also don't want to define two versions of CellML, because we need to be able to add to the "simple" version of CellML, without adding all the features defined in "hard" CellML. However some people are sure to get confused between levels and versions, and the potential to build up a vast 2-D array of levels and versions that software may be compliant to is frightening.

Maybe we can take a leaf from the Terminology section of the XML specification[1] and adapt their definition of **for interoperability**. Something like "Marks a sentence describing a non-binding recommendation included to increase the chances that CellML documents will be processed in a consistent manner". Reducing the strength of the Level 2 rules to recommendation status removes much of the confusion from the Levels/Versions system.

In fact we should probably consider inserting a terminology section in the introduction to the specification.

[1] http://www.w3.org/TR/2000/REC-xml-20001006#sec-terminology

4.1 Recommendation

Remove the current section on conformance levels from the specification. Insert the terminology section into the introduction in place of the **Definition of "*model*"** section. We should include definitions of "*model*" and any other potentially ambiguous terms that we can find in this section. Note that we currently have the definition of "*valid CellML document*" in this section.

4.1.1 Terminology

A model is an idealized representation of the rules that govern the behaviour of a system. The terms in the following list provide various useful classifications of model, some of which are used in the later definitions of the validity of CellML documents.

- **quantitative model**

  A model that represents the behaviour of a system mathematically.

- **qualitative model**

  A model that defines the relationships between objects in the system, without defining any mathematics that represent the behaviour of those objects.

- **complete quantitative model**

  A model that completely defines the mathematical representation of a system.

- **complete qualitative model**

  A model in which all objects of interest in a system are represented.

- **partial model**

  A model that contains the complete description of a portion of a model. Within that portion, the description is complete.

- **incomplete model**

  A model that is neither a complete qualitative model, a complete quantitative model, or a partial model. An incomplete model is generally a work in progress.

The terms defined in the following list are used in specifying the form of valid CellML documents, and the behaviour of CellML-conformant processing software.

- **may**

  Conforming CellML documents are permitted but not required to conform to the limitation described. Conforming CellML software is permitted but not required to behave as described if the suggested behaviour is relevant to the intended use of that software.

- **must**

  Conforming CellML documents must adhere to the limitation described. Conforming CellML software must behave as described if the suggested behaviour is relevant to the intended use of that software.

- **for interoperability**

  A non-binding recommendation included to increase the chances that CellML documents will be processed in a consistent manner.

- **error**

  A violation of the rules of this specification; results are undefined. Conforming CellML software may detect and report an error and may recover from it. It is recommended that software make information about errors available to the user.

- **valid CellML document**

  A document that conforms to all of the rules in this specification. A valid CellML document may describe a *complete*, *partial* or *incomplete* model.

- **valid CellML model**

  A valid CellML document that describes a *complete* model.

- **valid CellML subset document**

  A valid CellML document that only uses MathML elements from the CellML subset defined in Section ??.

- **CellML compliant software**

  CellML processing software that will interpret any valid CellML subset document according to the language semantics and processor rules defined in this specification.

- **fully MathML capable software**

  Software that can correctly interpret the full set of MathML content markup elements.

The definitions of **may** and **must** state that conformant CellML software need only concern itself with rules that are relevant to the intended use of that software. This means, for example, that software designed to render a model is not required to correctly interpret MathML.

4.1.2 Recommended handling of Level Two Rules

Each of the features listed in Section 3 can be handled in the specification in the following way.

- **Subsetting of MathML**

  - The contents of a **`<mathml:math>`** element must use only elements from the MathML content markup set.
  - For interoperability, CellML documents should use only elements from the CellML set of MathML elements (defined in Section ??) whenever possible. However, CellML documents may use other MathML elements if necessary.
  - CellML processing software must be able to correctly interpret all of the elements in the CellML basic set.
  - CellML software may interpret any elements from the MathML content set. Any MathML elements that are interpreted by the software must be interpreted according to the semantics defined by the MathML specification.

- **Function Calls**

  - For interoperability, where the **`<mathml:csymbol>`** element appears in a CellML document, it should conform to the following rules ...

- **Conversion Between Units Definitions**

  - For interoperability, software that can convert values between units definitions should achieve the same results as if the algorithm defined in Section ?? were used.

- **Equation Dimension Checking**

  - For interoperability, software that can check the dimensional consistency of equations should achieve the same result as if the algorithm defined in Section ?? were used. (This algorithm will define several situations in which an error may occur.)

Note the the “Subsetting of MathML” rules reflect a limitation rather than an extension to the specification, so a valid CellML document can contain any valid MathML content markup. As far as CellML documents are concerned, the CellML subset is only a recommendation, which is non-binding. However, CellML software is only required to interpret the elements in the CellML subset, which makes the interoperability recommendation pretty darn important.

This is a preferable option to stating something like “A valid CellML document must only contain elements from the CellML subset”, because this would effectively prevent people from extending MathML in a valid way. We don’t want to discourage people from doing this.

The algorithms for units conversion and equation dimension checking and all related examples and information can be moved to the appendices depending on popular demand.