Meeting Minutes 2 April 2001
MathML Subsetting Options

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

1 Introduction

In the 27 March meeting minutes[1], a system of subsetting the MathML content elements was proposed for use within CellML. In this proposal, there was a *basic* set of MathML elements that could be used to define any model that consisted of nothing but a system of ODEs, and numerous other sets of elements that were related to specific problem domains such as integration, partial differentiation, and imaginary numbers. Naturally Poul Nielsen, though abroad, did not waste any time shooting down this proposal.

In this document, we sum up Poul's comments, and present a summary of the principle approaches we can take to the subsetting problem, with their advantages and disadvantages.

2 Poul's Beef

The main problem that Poul had with Warren's MathML subsetting system was the sheer number of element sets that Warren had defined. He felt that this would lead to severe interoperability problems between CellML applications. For instance, if application A said it supported the *basic* and *integration* sets, and application B said it supported the *basic* and *imaginary* sets, you couldn't be sure that models exported from application A could be read by application B, even if the majority in fact could. Defining a number of sets makes things more difficult for both the user and the language designer (although that would be the first time he'd shown any concern for the language designer!)

Whereas Warren had created sets of elements based on their function, Poul suggested ranking elements purely on the basis of difficulty of implementation. "Easy" elements should be added to the *basic* set, even if the chances of them showing up in models was slim. This would hopefully not greatly affect the burden of MathML implementation. This would mean that elements related to trigonometry, for instance, which could be easily implemented, should be in the *basic* set.

Poul's single basic subset of MathML elements would contain the following of Warren's sets:

- *basic*
- *miscellaneous*, including **`<max>`**, **`<min>`**, **`<factorial>`**, **`<floor>`** and **`<ceiling>`**
- *integration*, needed for mechanics models like HMT
- *linear algebra*, needed for coupled systems of ODES ??
- *trigonometry*
- *constants and symbols*, including gems like **`<pi>`** and **`<imaginaryi>`**

[1]http://www.cellml.org/private/progress_reports/20010327_meeting_minutes.html

3 Subsetting Solutions

This section presents some of the possible solutions to the math subsetting problem.

3.1 Solution 1 : Warren's Original Proposal

Warren's original proposal is documented in the 27 March meeting minutes[2]. Warren's subsets consist of a *basic* set containing all of the elements necessary to describe models based on systems of ODEs, plus about 10 other sets of MathML elements, sorted by problem type.

The principal advantage of Warren's subsets was that it would be reasonably easy for software implementing CellML to implement the basic subset, which would cover the majority of pathway models as well as many simple EP models. Also, software would be able to easily specify what additional functionality it supported, and not be obliged to interpret large numbers of MathML elements that it didn't expect to use. This is particularly important for implementors of signal transduction pathway simulation packages, who would never achieve any level of CellML compliance if integration elements (which they are probably not interested in) are added to the *basic* set.

The principal disadvantage of using Warren's subsets is that it could easily create an interoperability nightmare where applications could export models in CellML that no other applications could read. Before MathML parsing could begin, applications would have to scan ahead looking for elements that they were unable to interpret. However, given the "math problem type" metadata elements in the CellML metadata specification, it should be possible for software to work out beforehand if it is likely to be able to do anything useful with a given model.

An alternative is for models and/or components to declare the subsets of MathML elements that make use of internally, possibly using some syntax in the CellML metadata namespace. This could be regarded as duplication of information that could be fairly easily evaluated by scanning the enclosed MathML.

3.2 Solution 2: Poul's Two Sets Proposal

As described in Section 2, Poul thought that Warren had defined too many sets. He preferred the definition of *basic* and *difficult* sets. The proposed contents of Poul's *basic* set are shown in Figure 1.

`<cn>`, **`<ci>`**, **`<apply>`**, **`<divide>`**, **`<minus>`**, **`<plus>`**, **`<times>`**, **`<power>`**, **`<root>`**, **`<abs>`**, **`<eq>`**, **`<neq>`**, **`<gt>`**, **`<lt>`**, **`<geq>`**,**`<leq>`**, **`<and>`**, **`<or>`**, **`<xor>`**, **`<not>`**, **`<diff>`**, **`<bvar>`**, **`<degree>`**, **`<piecewise>`**, **`<piece>`**, **`<otherwise>`**, **`<exp>`**, **`<log>`**, **`<ln>`**, **`<semantics>`**, **`<annotation>`**, **`<annotation-xml>`**, **`<quotient>`**, **`<factorial>`**, **`<max>`**, **`<min>`**, **`<rem>`**, **`<gcd>`**, **`<lcm>`**, **`<floor>`**, **`<ceiling>`**, **`<int>`**, **`<lowlimit>`**, **`<uplimit>`**, **`<sin>`**, **`<cos>`**, **`<tan>`**, **`<sec>`**, **`<csc>`**, **`<cot>`**, **`<sinh>`**, **`<cosh>`**, **`<tanh>`**, **`<sech>`**, **`<csch>`**, **`<coth>`**, **`<arcsin>`**, **`<arccos>`**, **`<arctan>`**, **`<arccosh>`**, **`<arccot>`**, **`<arccoth>`**, **`<arccsc>`**, **`<arccsch>`**, **`<arcsec>`**, **`<arcsech>`**, **`<arcsinh>`**, **`<arctanh>`**, **`<vector>`**, **`<matrix>`**, **`<matrixrow>`**, **`<determinant>`**, **`<transpose>`**, **`<selector>`**, **`<vectorproduct>`**, **`<scalarproduct>`**, **`<outerproduct>`**, **`<integers>`**, **`<reals>`**, **`<rationals>`**, **`<naturalnumbers>`**, **`<complexes>`**, **`<primes>`**, **`<exponentiale>`**, **`<imaginaryi>`**, **`<notanumber>`**, **`<true>`**, **`<false>`**, **`<emptyset>`**, **`<pi>`**, **`<eulergamma>`**, **`<infinity>`**

FIGURE 1: The MathML elements in Poul's proposed *basic* set.

[2] http://www.cellml.org/private/progress_reports/20010327_meeting_minutes.html

The advantage of having only two sets is that when an application says it implements "MathML in CellML Level One" (for want of a better name), you can be fairly confident that it will be able to exchange models with the vast majority of CellML compliant software (because this is all that most software will implement).

The disadvantage is that no-one really wants to have to implement the whole of the basic set. There are a number of elements in there that are difficult yet not useful for the majority of software (integration in particular), and many elements that are completely useless in general (think **`<primes>`**). It is likely that it no-one would ever really implement the complete *basic* set, making the interoperability argument a bit of a joke.

Furthermore, with two sets (the rest of the elements fall into "MathML in CellML Level Two"), extending the basic set becomes more problematic. If some hypothetical application is interested in spatial models, and they add support for Warren's *partial differentiation* element set, then they have no facility for advertising this capability (except by word of mouth), unless they implement the complete extended set. Realistically, this will never happen.

3.3 Solution 3 : Some Sort of Compromise

I can't believe I'm using the "C" word in meeting minutes. Let's look at what we'd like to have:

1. a very small number of MathML element sets
2. a *basic* set that isn't going to be hard for people to implement
3. a means by which software can advertise its handling of elements not in the *basic* set

Points 1 and 2 can be handled by merging some of Warren's sets.

The first step is to look for elements that can be added to the *basic* set. The *basic* set from the 27 March 2001 meeting minutes consisted only of the elements needed to define basic pathway and EP models, some logical expressions (for switching), and the MathML annotation elements (the preferred way of associating equation rendering information with an expression). Elements should only be added to the *basic* set if they are easy to implement and are not completely useless.

These elements include (and it could be argued that most of these are, in fact, largely useless):

- From the *miscellaneous arithmetic operators* set: **`<max>`**, **`<min>`**, **`<floor>`** and **`<ceiling>`**
- From the *trigonometry* set: everything
- From the *constants and symbols* set: **`<pi>`** and **`<notanumber>`**

The second step is to add more elements to the *not very useful* set. This set should only contain elements that are not likely to show up in any mainstream models whatsoever.

These elements include:

- From the *miscellaneous arithmetic operators* set: **`<quotient>`**, **`<factorial>`**, **`<rem>`**, **`<gcd>`** and **`<lcm>`**
- From the *theory of sets* set: everything
- From the *sequences and series* set: everything
- From the *statistics* set: everything
- From the *constants and symbols* set: everything apart from **`<pi>`** and **`<notanumber>`**

- *basic*

  `<cn>`, `<ci>`, `<apply>`, `<divide>`, `<minus>`, `<plus>`, `<times>`, `<power>`, `<root>`, `<abs>`, `<eq>`, `<neq>`, `<gt>`, `<lt>`, `<geq>`,`<leq>`, `<and>`, `<or>`, `<xor>`, `<not>`, `<diff>`, `<bvar>`, `<degree>`, `<piecewise>`, `<piece>`, `<otherwise>`, `<exp>`, `<log>`, `<ln>`, `<semantics>`, `<annotation>`, `<annotation-xml>`, `<max>`, `<min>`, `<floor>`, `<ceiling>`, `<sin>`, `<cos>`, `<tan>`, `<sec>`, `<csc>`, `<cot>`, `<sinh>`, `<cosh>`, `<tanh>`, `<sech>`, `<csch>`, `<coth>`, `<arcsin>`, `<arccos>`, `<arctan>`, `<arccosh>`, `<arccot>`, `<arccoth>`, `<arccsc>`, `<arccsch>`, `<arcsec>`, `<arcsech>`, `<arcsinh>`, `<arctanh>`, `<notanumber>`, `<pi>`

- *functions*

  `<declare>`, `<lambda>`, `<csymbol>`

- *integration*

  `<int>`, `<lowlimit>`, `<uplimit>`

- *partial differentiation*

  `<partialdiff>`, `<divergence>`, `<grad>`, `<curl>`, `<laplacian>`

- *imaginary*

  `<arg>`, `<real>`, `<imaginary>`, `<conjugate>`, `<sep>`

- *linear algebra*

  `<vector>`, `<matrix>`, `<matrixrow>`, `<determinant>`, `<transpose>`, `<selector>`, `<vectorproduct>`, `<scalarproduct>`, `<outerproduct>`

- *not very useful*

  `<equivalent>`, `<approx>`, `<factorof>`, `<implies>`, `<forall>`, `<exists>`, `<interval>`, `<inverse>`, `<condition>`, `<compose>`, `<ident>`, `<domain>`, `<codomain>`, `<image>`, `<domainofapplication>`, `<quotient>`, `<factorial>`, `<rem>`, `<gcd>`, `<lcm>`, `<set>`, `<list>`, `<union>`, `<intersect>`, `<in>`, `<notin>`, `<subset>`, `<prsubset>`, `<notsubset>`, `<notprsubset>`, `<setdiff>`, `<card>`, `<cartesianproduct>`, `<sum>`, `<product>`, `<limit>`, `<tendsto>`, `<mean>`, `<sdev>`, `<variance>`, `<median>`, `<mode>`, `<moment>`, `<momentabout>`, `<integers>`, `<reals>`, `<rationals>`, `<naturalnumbers>`, `<complexes>`, `<primes>`, `<exponentiale>`, `<imaginaryi>`, `<true>`, `<false>`, `<emptyset>`, `<eulergamma>`, `<infinity>`

FIGURE 2: A second stab at a complete subsetting of the MathML content elements for use in CellML.

We are now left with the sets of MathML elements shown in Figure 2. It is anticipated that elements may be moved from the *not very useful* set into other sets in the future, as the needs for CellML's mathematical capabilities develop. For instance, elements like **`<mean>`** and **`<variance>`** may form the basis of a stochastic set.

The third of our requirements for an ideal scheme was a simple means for software to advertise its "MathML in CellML" compliance. The basic premises of MathML compliance are:

1. Software can be CellML-compliant without knowing anything about MathML. That means that software that is only interested in rendering can just ignore equations and concentrate on elements in the CellML namespace. Similarly, a tool that takes a CellML document and puts it in a database might only care about elements in the CellML and CellML metadata namespaces.
2. For software that is concerned with mathematics, we define some levels of compliance called "MathML in CellML".
3. In order to be able to claim any level of "MathML in CellML" compliance, software must be able to correctly interpret all of the elements in the *basic* set. It does not have to export them, or provide facilities for modellers to view or create mathematics involving these elements, but it must be able to **interpret** them.
4. Software knows what elements it can interpret. The CellML specification will not recommend a course of action for software that receives CellML documents containing MathML elements that it is unable to interpret.

Some options for defining further levels of "MathML in CellML" compliance are:

1. **All or Nothing**: (More accurately, "All or Some"). We throw away all of the sets other than the *basic* set, and software can advertise only support for the MathML elements in the basic set, or full MathML support.
2. **Element by Element**: We throw away all of the sets other than the *basic* set. Software can advertise basic support, plus a list of additional MathML elements that it can interpret.
3. **Compliance Sets**: Compliance is based on the sets proposed in Figure 2. Minimally compliant software can advertise "MathML in CellML : basic" compliance. If software also can correctly interpret all of the elements in the *functions* and *integration* sets, for instance, it can advertise "MathML in CellML : basic, functions & integration" compliance.