Progress Report 27 March 2001
Subsectioning of MathML for use in CellML

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributor:
Melanie Nelson (Physiome Sciences Inc.)

1 Introduction

This document proposes a scheme/some schemes for subsectioning the MathML element set for use in CellML. Why? One of the most frequent complaints about CellML is having to deal with the full complexity of MathML. If the MathML element set were divided up into some logical subsets, then software could properly advertise that it was only able to handle certain sets of elements or certain problem types.

Throughout this document, when we refer to "MathML elements", we really mean just the MathML content elements. The CellML specification is not concerned with the presentation form of MathML markup, although this could be conceivably used for describing renderings of CellML variables and equations — such rendering information is considered metadata. Also, elements that are marked as deprecated in the MathML 2.0 specification[1] will not be considered.

2 Problem or Element Type

The most important issue to consider is the basis of any element subsetting. It is most likely that software will be able to deal with specific types of modelling problems. An example of a problem type is "system of ordinary differential equations". It should be possible to define some common problem types and associate a subset of MathML elements with each problem type. Software that claimed to deal with that problem type would be expected to read in valid MathML using the element set associated with that problem type, and would only output MathML elements from that element set for a problem of the specified type. Possible problem types are discussed further in Section 3.

An alternative would be to split up the MathML element set by element type (i.e., function). Software could claim to be able to handle elements from a number of element sets. A specific problem type would generally involve using several sets. Possible element sets and their relationships to problem types are described in Section 4.

3 Problem Types

The American National Institute of Science and Technology has assembled a repository of links to mathematical software sorted by problem category. Their Guide to Available Mathematical Software[2] website defines a problem decision tree, the root categories of which are given below.

- A — Arithmetic, error analysis
- B — Number theory
- C — Elementary and special functions (search also class L5)
- D — Linear Algebra

[1] http://www.w3.org/TR/2001/REC-MathML2-20010221
[2] http://gams.nist.gov/

- E — Interpolation
- F — Solution of nonlinear equations
- G — Optimization (search also classes K, L8)
- H — Differentiation, integration
- I — Differential and integral equations
- J — Integral transforms
- K — Approximation (search also class L8)
- L — Statistics, probability
- M — Simulation, stochastic modeling (search also classes L6 and L10)
- N — Data handling (search also class L2)
- O — Symbolic computation
- P — Computational geometry (search also classes G and Q)
- Q — Graphics (search also class L3)
- R — Service routines
- S — Software development tools
- Z — Other

With the exception of categories B, O, P, Q, R, S and Z, all of the categories contain some subcategories which might conceivably be relevant to CellML and some that probably aren't. The task of analysing the hundreds of subcategories and defining sets of MathML elements that are appropriate for the problem type is too vast to contemplate.

4 Element Types

We will consider how the MathML element set might be divided up into element types. The MathML specification itself divides the MathML element set into a number of element types in the complete listing of content elements given in Section 4.4 of the spec[3]. The categories are given below:

- token elements
- basic content elements
- arithmetic, algebra and logic
- relations
- calculus and vector calculus
- theory of sets
- sequences and series
- elementary classical functions
- statistics
- linear algebra
- semantic mapping elements

[3] http://www.w3.org/TR/2001/REC-MathML2-20010221/chapter4.html#contm_elem

- constant and symbol elements

This subsetting does not suit our needs because, in most cases, each category contains elements that are useful for specific problems and elements that aren't. For instance, the "basic content elements" category contains the **<apply>** element which is the basis of any equation, but also elements like **<declare>**, which could be considered elements useful for extending the basic functionality of MathML.

4.1 Lets Jump In

As the eminently quotable Chris Bradley would say: "If you want to do it right, you've got to do it yourself". Having fruitlessly tried to steal the work of others (one of the fundamental principles of CellML development), I decided to just have a go. A good starting point is to look at the most basic lumped parameter electro-physiological models — for instance, the Hodgkin Huxley squid axon model of 1952. The MathML elements used in this are listed in Figure 1,

---

- **token elements**
  <cn>, **<ci>**
- **basic content elements**
  <apply>
- **arithmetic, algebra and logic**
  <divide>, **<minus>**, **<plus>**, **<times>**
- **relations**
  <eq>
- **calculus**
  <diff>, **<bvar>**
- **elementary classical functions**
  <exp>

FIGURE 1: The complete set of MathML content markup elements needed to defi ne the governing equations from the Hodgkin Huxley squid axon model of 1952. These will form the basis of the *basic* MathML elements.

The set of elements from the H-H model is actually quite a useful set. There are some obvious issues. First of all, if we require the recognition of **<exp>**, we should really require the recognition of **<ln>** and **<log>**, which are also in MathML's "elementary classical functions" subset. In fact, if you look at the rest of the elements in that set, you see that they are all trigonometric functions. In terms of implementation, most of these map simply to functions built into any programming language, or can be expressed in terms of exponential functions. These elements could be incorporated into our *basic* set. However, as they are almost never used in common cellular models, it makes sense to just place them in their own set called *trigonometry*.

The basic set also lacks some of the elements needed to do proper differentiation. In addition to the **<diff>** and **<bvar>** elements, we also need the **<degree>** element, which is needed to do second derivatives. Let's consider the rest of the elements in MathML's calculus subset. The first question is:

do we want integration to be in the basic set? Probably not. Let's define an *integration* set consisting of the **<int>**, **<lowlimit>** and **<uplimit>** elements. The rest of the elements then fall in the *partial differentiation* set, which consists of the **<partialdiff>**, **<divergence>**, **<grad>**, **<curl>** and **<laplacian>** elements. (Maybe we actually want to separate out partial differentiation from the other elements.)

Moving to more complex issues now: another model we've looked at in detail is the Luo Rudy model from 1994. This model makes use of switching functionality in many of its gate variable calculations. These equations are analogous to the common programming form `a=(V>0)?x:y`. MathML 2.0 introduced three new elements to deal with this kind of behaviour and these should be added to the basic set. They are the **<piecewise>**, **<piece>** and **<otherwise>** elements. In order to make these elements useful, we will need some comparison operators from the "relations" subset defined by MathML. We should definitely make use of **<neq>**, **<gt>**, **<lt>**, **<geq>** and **<leq>**. The remainder of the operators in this set can be filed in the *really not useful* set of elements.

We have only the four basic arithmetic operators in the basic set so far, plus three classical functions. We should probably add the **<power>** and **<root>** elements to the basic set, as powers appear in many basic models and these are easily implemented in software. The remainder of the elements in the "arithmetic, algebra and logic" set defined by MathML are either uncommon arithmetic operators, involved in logic and set theory, or to do with imaginary numbers. We'll consider how these can be divided up later.

To quickly bin some other elements: CellML can borrow MathML's "theory of sets", "sequences and series", "statistics", "linear algebra" and "constant and symbol elements" sets. These are largely useless to CellML, but we may as well keep them differentiated to make them more usable.

The "semantic mapping elements" is a more tricky proposition, because it defines the elements needed to annotate an equation with its rendering (if we chose to do it using the MathML mechanism, rather than a CellML metadata mechanism). There are only three, and two of those can only be used in conjunction with the other. Basically, if software knows how to deal with the **<semantics>** element, then it knows enough to be able to ignore the **<annotation>** and **<annotation-xml>** elements. Because some MathML generation software might well make use of this fairly often, and it has a low overhead, we may as well add it to the basic set.

The hard part now is deciding how to split up the "basic content elements" set. This includes set-related elements, function related elements, and domain-related elements. Let's eliminate the function related elements first. In order to declare a function defined in MathML, we need the **<declare>** and **<lambda>** elements. In order to call that function, or call functions declared in other languages, we need the **<csymbol>** element. These elements form a tidy *function* set.

5 Recommendation

We are now in a position to start defining some sets. These sets are in the subsequent figures. It is expected that some sets could be merged to get a smaller number of larger sets.

<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>**

FIGURE 2: The *basic* set.

<declare>, **<lambda>**, **<csymbol>**

FIGURE 3: The *functions* set.

<quotient>, **<factorial>**, **<max>**, **<min>**, **<rem>**, **<gcd>**, **<lcm>**, **<floor>**, **<ceiling>**

FIGURE 4: The *miscellaneous arithmetic operators* set.

<int>, **<lowlimit>**, **<uplimit>**

FIGURE 5: The *integration* set.

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

FIGURE 6: The *partial differentiation* set.

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

FIGURE 7: The *imaginary* set.

<sin>, **<cos>**, **<tan>**, **<sec>**, **<csc>**, **<cot>**, **<sinh>**, **<cosh>**, **<tanh>**, **<sech>**, **<csch>**, **<coth>**, **<arcsin>**, **<arccos>**, **<arctan>**, **<arccosh>**, **<arccot>**, **<arccoth>**, **<arccsc>**, **<arccsch>**, **<arcsec>**, **<arcsech>**, **<arcsinh>**, **<arctanh>**,

FIGURE 8: The *trigonometry* set.

<set>, **<list>**, **<union>**, **<intersect>**, **<in>**, **<notin>**, **<subset>**, **<prsubset>**, **<notsubset>**, **<notprsubset>**, **<setdiff>**, **<card>**, **<cartesianproduct>**

FIGURE 9: The *theory of sets* set.

<sum>, **<product>**, **<limit>**, **<tendsto>**

FIGURE 10: The *sequences and series* set.

<mean>, **<sdev>**, **<variance>**, **<median>**, **<mode>**, **<moment>**, **<momentabout>**

FIGURE 11: The *statistics* set.

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

FIGURE 12: The *linear algebra* set.

<integers>, **<reals>**, **<rationals>**, **<naturalnumbers>**, **<complexes>**, **<primes>**, **<exponentiale>**, **<imaginaryi>**, **<notanumber>**, **<true>**, **<false>**, **<emptyset>**, **<pi>**, **<eulergamma>**, **<infinity>**

FIGURE 13: The *constants and symbols* set.

<equivalent>, **<approx>**, **<factorof>**, **<implies>**, **<forall>**, **<exists>**, **<interval>**, **<inverse>**, **<condition>**, **<compose>**, **<ident>**, **<domain>**, **<codomain>**, **<image>**, **<domainofapplication>**

FIGURE 14: The *not very useful* set.