CellML Specification

Final Draft — 18 May 2001

4 Mathematics

4.1 Introduction

CellML is aimed at unambiguously specifying the underlying mathematics of a cellular model. Model components may contain mathematical expressions that manipulate the values of variables that belong to them. These expressions are also free to use (but must not modify) the values of any other variable declared in those components.

Mathematical expressions are embedded in CellML documents using Mathematical Markup Language (MathML)[1], an XML-based language that encodes the underlying structure of a mathematical expression. The first version[2] of MathML reached recommendation status at the World Wide Web Consortium (W3C)[3] in April 1998. A second version[4], which extended the functionality of MathML 1.0 in many key areas, was released in February 2001. The MathML 2.0 Recommendation also deprecates the use of some MathML 1.0 elements. CellML uses a subset of the elements from MathML 2.0, known as the *content markup* element set, which includes several deprecated elements from MathML 1.0.

CellML 1.0 does not require processing software to implement support for scripting. If software chooses to do so, some recommendations on the use of scripting are given in Appendix B[5].

4.2 Basic Structure

4.2.1 Definition of mathematics

All mathematical expressions defined using MathML must be placed inside a **<mathml:math>** element. A **<mathml:math>** element must only be defined in **<cellml:component>** or **<cellml:role>** elements. The **mathml** and **cellml** namespace prefixes are used throughout this section to indicate that elements are in the MathML and CellML namespaces, respectively. The **<cellml:role>** and **<cellml:variable_ref>** elements mentioned in this section are first introduced in Section 7[6] of this specification.

<mathml:math> elements that occur as child elements of **<cellml:component>** elements can be used to define arbitrary expressions in terms of the variables declared in that component. A mathematical expression may make use of any variable declared within the current component by placing the variable's name within a **<mathml:ci>** element. Expressions must only modify the values of variables that *belong* to that component. Variables that belong to a component are those that are not declared with a **public_interface** or **private_interface** attribute value of "in".

<mathml:math> elements that occur as child elements of **<cellml:role>** elements (these are defined within **<cellml:variable_ref>** elements, which are in turn defined within **<cellml:reaction>** elements) can be used to define expressions that modify the values of specific variables in specific ways.

[1] http://www.w3.org/Math
[2] http://www.w3.org/TR/1998/REC-MathML-19980407
[3] http://www.w3.org/
[4] http://www.w3.org/TR/2001/REC-MathML2-20010221
[5] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_math_scripting_functionality_in_cellml
[6] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_reactions

These expressions may make use of any variable declared in the current component but must only modify the value of the variable referenced by the ancestor **<cellml:variable_ref>** element, subject to further limitations that are described in Section 7.2[7].

CellML processing software must interpret MathML elements according to the semantics defined in the MathML 2.0 Recommendation[8]. However, CellML 1.0 does define some restrictions on, and additions to, the MathML syntax. These are covered in the subsequent sections.

4.2.2 MathML's presentation and content markup elements

The complete set of elements defined in the MathML 2.0 Recommendation[9] is split into two principal subvocabularies: the *presentation markup* and *content markup* elements. The presentation markup elements are oriented towards describing the visual rendering of mathematical expressions and objects. The content markup elements aim to capture the underlying meaning of a mathematical expression or object, without regard to its presentation.

CellML is used to describe the structure and mathematics of cellular models. For this reason, valid CellML documents must only contain content markup elements within a **<mathml:math>** element. There is one exception: model authors may use MathML presentation markup elements inside a **<mathml:annotation** element to associate rendering information with a particular expression. CellML processing software may ignore the contents of **<mathml:annotation>** and **<mathml:annotation-xml>** elements.

An example demonstrating the embedding of MathML content and presentation markup elements in a CellML document is presented in Section 4.3.

4.2.3 The CellML subset of MathML content elements

Valid CellML documents may contain any MathML content markup elements within a **<mathml:math>** element, as long as the arrangement of these elements follow the rules defined in the MathML 2.0 Recommendation[10]. However, it is anticipated that it will be some time before software is able to interpret all of these elements. To encourage interoperability, this section defines a subset of the MathML content markup elements known as the *CellML subset*. CellML documents that only contain content markup elements from the CellML subset are known as *valid CellML subset documents*. CellML processing software may only call itself *CellML compliant* if it is able to correctly interpret all of the MathML elements in the CellML subset according to the semantics defined in the MathML 2.0 Recommendation.

The complete list of MathML elements in the CellML subset is given in Figure 5. Many of the elements in the CellML subset are included to provide facilities for the definition of algebraic and ordinary differential equations. Others (such as the trigonometric operators) have been included because they are reasonably straightforward to translate to computer code.

4.2.4 Associating units with numbers

To ensure that models are robust and portable, all variables and numbers that occur in mathematical expressions within a CellML document must have units associated with them. CellML's units framework is introduced in Section 5[11] and the association of units with variables is presented in Section 3.2.3[12]. The association of units with numbers in equations requires an extension to MathML. This can be done in a manner consistent with the association of units with variables and with application-specific extensions to

[7] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_rxn_basic_structure
[8] http://www.w3.org/TR/2001/REC-MathML2-20010221
[9] http://www.w3.org/TR/2001/REC-MathML2-20010221
[10] http://www.w3.org/TR/2001/REC-MathML2-20010221
[11] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_units
[12] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_structure_variable_element

- ***token elements***

  <cn>, **<ci>**
- ***basic content elements***

  <apply>, **<piecewise>**, **<piece>**, **<otherwise>**
- ***relational operators***

  <eq>, **<neq>**, **<gt>**, **<lt>**, **<geq>**, **<leq>**
- ***arithmetic operators***

  <plus>, **<minus>**, **<times>**, **<divide>**, **<power>**, **<root>**, **<abs>**, **<exp>**, **<ln>**, **<log>**, **<floor>**, **<ceiling>**, **<factorial>**
- ***logical operators***

  <and>, **<or>**, **<xor>**, **<not>**
- ***calculus elements***

  <diff>
- ***qualifier elements***

  <degree>, **<bvar>**, **<logbase>**
- ***trigonometric operators***

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

  <true>, **<false>**, **<notanumber>**, **<pi>**, **<infinity>**, **<exponentiale>**
- ***semantics and annotation elements***

  <semantics>, **<annotation>**, **<annotation-xml>**

FIGURE 5: The *CellML subset* of MathML content markup elements, grouped according to function. All elements in this figure are in the MathML namespace.

CellML by adding a **units** attribute in the CellML namespace to the **<mathml:cn>** element, which encloses all numbers. The example presented in Section 4.3 demonstrates this.

4.2.5 Definition of scripts

CellML 1.0 does not define any means with which model authors can embed scripts in CellML documents in a portable way. It is anticipated that this functionality will be defined in a subsequent version of CellML. The use of scripts in CellML is strongly discouraged. CellML is aimed at specifying a model in terms of its most basic governing equations. Wherever possible, mathematical equations should be used to specify the changing behaviour of a model's state variables.

If implementors do decide to add scripting functionality to CellML documents, these scripts must be defined within elements placed in an application-specific extension namespace. Implementors are advised to follow the recommendations on the best practices for embedding and executing scripts described in Appendix B[13]. The key recommendations are summarised in the following list:

- For interoperability, scripts should be defined using ECMAScript.
- The **<mathml:csymbol>** element should be used from within MathML markup to call scripts defined using a non-MathML syntax. These elements must define a **definitionURL** that identifies the element containing the script, and an **encoding** attribute specifying the scripting language used.
- Function names (or the identifier used to reference a script) should be valid CellML identifiers, as defined in Section 2.2.1[14].
- The content of a **<mathml:csymbol>** element should be a human-readable identifier for the script, preferably the function name.
- Scripts should be side-effect free. That is, scripts should not alter the values of their arguments.

4.3 Examples

The CellML fragment in Figure 6 demonstrates how MathML can be employed within CellML to define mathematical expressions. This fragment is part of the definition of a component that represents the behaviour of the *n* gate from the potassium channel in the Hodgkin-Huxley squid axon model of 1952. The component contains two units definitions (with syntax defined in Section 5[15]), two variable declarations (with syntax defined in Section 3[16]), and a block of MathML that defines an expression calculating the *alpha* variable of the *n* gate as well as the rendering of this equation, which is given in Equation (2).

$$alpha\_n = 1.0\frac{0.01(V + 10.0)}{\exp(0.1(V + 10.0)) - 1.0} \tag{2}$$

Content that isn't defined using the MathML content markup elements can be associated with a MathML expression using the **<mathml:semantics>**, **<mathml:annotation>** and **<mathml:annotation-xml>** elements. The first child of a **<mathml:semantics>** element is the expression to be annotated, and the subsequent **<mathml:annotation>** and **<mathml:annotation-xml>** elements contain character data and XML annotations, respectively. In the CellML fragment in Figure 6, the expression of interest has been annotated with rendering information encoded using the MathML presentation markup elements. The MathML presentation elements are very flexible and it is possible to produce the same rendering of an equation in many ways — the choice of elements in Figure 6 is somewhat arbitrary.

[13] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_math_scripting_functionality_in_cellml
[14] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_fundamentals_identifiers
[15] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_units
[16] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_model_structure

```
<component
    name="potassium_channel_n_gate"
    xmlns="http://www.cellml.org/cellml/1.0#"
    xmlns:cellml="http://www.cellml.org/cellml/1.0#"
    xmlns:cmeta="http://www.cellml.org/metadata/1.0#">

  <units name="per_millisecond">
    <unit prefix="milli" units="second" exponent="-1" />
  </units>
  <units name="millivolt">
    <unit prefix="milli" units="volt" />
  </units>
  <units name="per_millivolt">
    <unit prefix="milli" units="volt" exponent="-1" />
  </units>

  <variable name="alpha_n" units="per_millisecond" />
  <variable name="V" public_interface="in" units="millivolt" />

  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <semantics>
      <apply id="alpha_n_calculation"><eq />
        <ci> alpha_n </ci>
        <apply><times />
          <cn cellml:units="per_millisecond"> 1.0 </cn>
          <apply><divide />
            <apply><times />
              <cn cellml:units="per_millivolt"> 0.01 </cn>
              <apply><plus />
                <ci> V </ci>
                <cn cellml:units="millivolt"> 10.0 </cn>
              </apply>
            </apply>
            <apply><minus />
              <apply><exp />
                <apply><times />
                  <cn cellml:units="per_millivolt"> 0.1 </cn>
                  <apply><plus />
                    <ci> V </ci>
                    <cn cellml:units="millivolt"> 10.0 </cn>
                  </apply>
                </apply>
              </apply>
              <cn cellml:units="dimensionless"> 1.0 </cn>
            </apply>
          </apply>
        </apply>
      </apply>
      <annotation-xml encoding="MathML-Presentation">
        <mrow>
          <mi> alpha_n </mi><mo> = </mo><mn> 1.0 </mn>
          <mfrac>
            <mrow>
              <mn> 0.01 </mn><mo> ( </mo>
              <mi> V </mi><mo> + </mo><mn> 10.0 </mn>
              <mo> ) </mo>
            </mrow>
            <mrow>
              <mo>exp</mo>
              <mo> ( </mo><mn> 0.1 </mn><mo> ( </mo>
              <mi> V </mi><mo> + </mo><mn> 10.0 </mn>
              <mo> ) </mo><mo> ) </mo>
              <mo> - </mo>
              <mn> 1.0 </mn>
            </mrow>
          </mfrac>
        </mrow>
      </annotation-xml>
    </semantics>
  </math>
</component>
```

FIGURE 6: Part of the definition of a component that represents the behaviour of the *n* gate from the potassium channel in the Hodgkin-Huxley squid axon model of 1952. See text for more details.

The **<mathml:apply>** element at the top level of the expression defines an **id** attribute, which can be used to associate further metadata with the expression. The linking of metadata with elements in a CellML document is described in more detail in Section 8.2[17].

All of the **<mathml:cn>** elements in the equation define **cellml:units** attributes, which associate a units definition with the number delimited by the **<mathml:cn>** element. The inclusion of units in the equation allows CellML processing software to check that the dimensions of the terms in an equation are consistent, as discussed in Section 5[18]. The presence of the unit scale factor on the right hand side of the equation is needed for the equation to have consistent dimensions.

4.4 Rules for CellML Documents

4.4.1 The <mathml:math> element

1. **Allowed use of the <mathml:math> element**

   - The **<mathml:math>** element must only appear as a child of **<cellml:component>** or **<cellml:role>** elements.

     [In this and subsequent rules, the use of the **mathml** and **cellml** namespace prefixes indicates that elements and attributes are in the MathML and CellML namespaces, respectively.]

   - All elements in the MathML namespace that are within a **<mathml:math>** element, and not within a **<mathml:annotation>** or **<mathml:annotation-xml>** element, must be taken from the complete set of MathML content markup elements, as defined in Section 4.4 of the MathML 2.0 Recommendation[19], with the addition of the **<mathml:logbase>** element.

     [CellML only makes use of the content markup elements from MathML. However presentation markup elements may be used within the annotation elements to associate rendering information with expressions. For interoperability, it is recommended that only a subset of the MathML content markup elements be used, as discussed in Section 4.2.3. The **<mathml:logbase>** element was erroneously omitted from the list of content markup elements in Section 4.4 of the MathML 2.0 Recommendation.]

   - The contents of a **<mathml:math>** element must conform to the MathML 2.0 Recommendation[20] from the W3C.

4.4.2 The <mathml:ci> element

1. **Allowed use of the <mathml:ci> element**

   - After leading and trailing whitespace is removed, the content of a **<mathml:ci>** element must match the value of the name of a variable declared within the current component.

     [The **<mathml:ci>** element is used to reference variables from inside equations. Whitespace may be added before and/or after a variable's name to make the MathML more readable. The handling of whitespace in MathML is described in more detail in Section 2.4.6 of the MathML 2.0 Recommendation[21].]

[17] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_metadata_basic_structure
[18] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_units
[19] http://www.w3.org/TR/2001/REC-MathML2-20010221/chapter4.html#contm_elem
[20] http://www.w3.org/TR/2001/REC-MathML2-20010221
[21] http://www.w3.org/TR/2001/REC-MathML2-20010221/chapter2.html#fund_collapse

4.4.3 The `<mathml:cn>` element

1. **Allowed use of the `<mathml:cn>` element**

   - A **`<mathml:cn>`** element must define a **`cellml:units`** attribute.

     [All bare numbers in MathML content markup are enclosed in a **`<mathml:cn>`** element in the MathML namespace.]

2. **Allowed values of the `cellml:units` attribute**

   - The value of the **`cellml:units`** attribute must be taken from the standard dictionary of units given in Section 5.2.1[22], or be the value of the **`name`** attribute on a **`<units>`** element defined in the current **`<component>`** or **`<model>`** element.

4.4.4 Modification of variables

- A mathematical expression defined using MathML must only modify the values of variables that belong to the current component.

  [Variables that belong to a component are those that are not declared with a **`public_interface`** or **`private_interface`** attribute value of `"in"`.]

4.5 Rules for Processor Behaviour

4.5.1 Ordering of expressions

CellML processing software must not assume that the ordering of expressions within a CellML document has any significance.

4.5.2 Treatment of annotations

CellML processing software may ignore the contents of **`<mathml:annotation>`** and **`<mathml:annotation`** elements. CellML processing software must assume that the contents of the first child of a **`<mathml:semantics>`** element define an expression describing the mathematical behaviour of the model.

[22] http://www.cellml.org/public/specification/20010518/cellml_specification.html#sec_units_cellml_units_dictionary