Meeting Minutes 15 February 2001

Authors:
Warren Hedley (Bioengineering Institute, University of Auckland)
Melanie Nelson (Physiome Sciences Inc.)
Contributors:
Poul Nielsen (Bioengineering Institute, University of Auckland)
Chris Penland (Physiome Sciences Inc.)

1 Introduction

Further reformattings and close readings of the reactions and units parts of the CellML specification caused some concerns to be raised, as described in this document. The document was then run by Chris Penland at Physiome and Poul Nielsen in Auckland for more input, and then extended further. The original version has been archived for reference.

2 Further Concerns with Reactions

We are in the process of formalizing the rules for the use of the **<reaction>** element, as well as the expectations for processor behaviour with respect to the **<reaction>** element. This process has highlighted some unresolved issues regarding reactions, which are discussed below.

2.1 The need for arbitrary math to relate the general rate and the rate of change of specific species

There has been a lot of discussion recently about whether or not it is necessary to allow modellers to use arbitrary mathematical relationships between the general reaction rate and the rate of change of a particular species in conjunction with the **<reaction>** element. Everyone agreed that modellers would be able to define such relationships by using a **<math>** element outside of the **<reaction>** element (but still inside the same **<component>** element). The question was whether it is useful to allow the **<math>** element to go inside the **<variable_ref>** element that refers to the variable representing the species whose rate of concentration change is defined by the mathematics.

No one could come up with an example of a case where the simple relationship between the general reaction rate and the rate of change of a particular species that is implied by the **delta_variable** and **stoichiometry** attributes on the **<role>** element would not be correct and sufficient. However, no one could rule out such a situation, either. Poul argued that since we couldn't think of any examples where we'd need the arbitrary math, we should not allow it to be included. He thought that we should not allow "features" into CellML unless they either supported a demonstrated modelling need or simplified the language. Warren and Melanie never really thought of supporting arbitrary math in the **<reaction>** element for the relationship between the reaction rate and the rate of change of a species as an extra "feature", since the modeller would be able to define such math outside of the **<reaction>** element, and we already allow math inside the **<reaction>** element for the kinetic rate law. However, Poul's argument that it is easier to add new features later than to take out existing features was a good one, so the issue was put back on the table for discussion.

After much soul searching, Melanie finally came up with an example of a case in which a modeller would want to be able to define the relationship between the general reaction rate and the rate of change of a species using MathML, rather than using the **delta_variable** and **stoichiometry** attributes. If the stoichiometry of a reaction is not known, a modeller would not want to include a value for the **stoichiometry** attribute. Doing so would imply more than just a mathematical relationship between

the rate variable and the variable representing the rate of change of the species (the *delta* variable). It would also imply chemical knowledge of the stoichiometry of the reaction, since the **stoichiometry** attribute is also used to support the required capability of representing the chemical reaction expressions. This would be a very bad situation: we would be forcing a modeller to include information that he or she knows to be incorrect (or at least unsupported by the available evidence) in the model. This situation could be avoided by allowing the modeller to define a MathML expression relating the general reaction rate and the rate of change of the species.

Another, related situation in which it would be useful to be able to define a MathML expression for the relationship between the rate variable and the delta variable would be a model in which the stoichiometry of the species is a variable. The modeller might be trying to examine several possible values for the stoichiometry, to determine which one fits best with the rest of the information in the model. This could only be supported by allowing the modeller to use MathML to define the relationship between the rate variable, the delta variable, and a variable representing the stoichiometry.

Of course, these examples could be supported simply by requiring that any such mathematics be placed outside of the reaction element. However, Warren and Melanie feel that it is useful to allow the modeller to indicate which variable is representing the rate of change of the chemical species (the delta variable). They believe that it is important to allow the modeller to make this indication because software might use this information to present the mathematics relating the rate variable and the delta variable in meaningful context. For instance, it might group all of the relationships between the rate variable and the delta variables for all of the reactants and products, or it might display these equations in a different color.

2.2 What does it mean to include <math> elements in a <reaction> element?

This question is closely related to the issue of whether or not we should allow **<math>** elements in **<variable_ref>** elements with roles of "product" or "reactant" as well as in **<variable_ref>** elements with roles of "rate".

The meaning of a **<math>** element occurring inside a **<variable_ref>** element with a **<role>** element on which the **role** attribute value is "rate" is reasonably clear: the equations in the math element define the kinetic rate law (i.e., they set the value of the referenced variable, which represents the general reaction rate).

The meaning of a **<math>** element occurring inside a **<variable_ref>** element with a **<role>** element on which the **role** attribute value is "reactant" or "product" is less obvious. We presently assume that this math should define the relationship between the general reaction rate and the rate of change in concentration of the referenced species.

The following questions must be answered:

- Is it reasonable to have a **<math>** element in a **<variable_ref>** element mean different things based on the value of the **role** attribute on the enclosed **<role>** element? (We could instead include the **<math>** element in the **<role>** element itself, as discussed in Section 2.3. However, doing so would not resolve this question.)
- Is it reasonable to outlaw the inclusion of **<math>** elements in **<variable_ref>** elements with other values of the **role** attribute (i.e., "catalyst", "activator", "inhibitor", or "modifier")? The meaning of mathematics associated with variables fulfilling these roles has not been previously discussed. However, it would be reasonable to use a **<math>** element inside a **<variable_ref>** element with on of these roles to calculate an intermediate variable used in the rate calculation that is dependent on the referenced variable. For instance, a modeller might calculate an effective concentration of a catalyst based on a variable, such as pH or temperature.
- Should we instead outlaw the inclusion of **<math>** elements in **<variable_ref>** elements with any value of the **role** attribute other than "rate"? Modellers could still define an arbitrary math-

ematical function relating the general reaction rate and the rate of concentration for a species (the equations would simply be stored in a **<math>** element in the **<component>** element that contains the **<reaction>** element). However, there would no longer be a way for the modeller to indicate to processing software that a given equation or set of equations was performing this function (see Section 2.4 for another possible method of providing this information). If software knows which equations set the rate of concentration change of a species it may be able to present those equations to the modeller in a more meaningful context.

2.3 Where should the <math> element be placed in the <reaction> element?

Sadly, our current examples and the current rules disagree on the question of where the **<math>** element should be placed in the **<reaction>** element. The examples have the **<math>** element inside the **<variable_ref>** element, and the rules state that the **<math>** element should appear inside the **<role>** element. These two possibilities are shown in Figure 1.

```
<!-- mathematics in the <variable_ref> element -->
<reaction reversible="yes">
  <variable_ref variable="A">
    <role
        role="reactant" direction="forward"
        delta_variable="delta_A" stoichiometry="1" />
  </variable_ref>
  ...
  <variable_ref variable="r">
    <role role="rate" />
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <!-- kinetic rate law -->
    </math>
  </variable_ref>
</reaction>

<!-- mathematics in the <role> element -->
<reaction reversible="yes">
  <variable_ref variable="A">
    <role
        role="reactant" direction="forward"
        delta_variable="delta_A" stoichiometry="1" />
  </variable_ref>
  ...
  <variable_ref variable="r">
    <role role="rate">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <!-- kinetic rate law -->
      </math>
    </role>
  </variable_ref>
</reaction>
```

FIGURE 1: The two possible location for **<math>** elements within reaction elements are shown, with the mathematics themselves omitted.

Warren favors putting the **<math>** element inside the **<role>** element. Melanie can't come up with

any reason not to do this, other than that it requires her to change all of the reaction model examples *again*. She would like this issue to be thought about carefully, to minimize the chance that she'll have to change the examples again next week. Here is what each possible location for the **<math>** element implies:

- Putting the math element inside the **<variable_ref>** element indicates that the equations are some way relevant to the referenced variable, regardless of its role.
- Putting the **<math>** element inside the **<role>** element indicates that the equations are in some way relevant to the referenced variable in that specific role. This has the added benefit of making it a little more logical to limit the inclusion of **<math>** elements to only certain values of the **<role>** attribute, since the limiting attribute is on the parent element for the **<math>** element.

A **<variable_ref>** element that has a **<role>** element with a **role** attribute value of `"rate"` cannot have any other **<role>** elements. Therefore, the implications of the positioning of the **<math>** element are only relevant for **<role>** elements with a **role** attribute value of `"product"` or `"reactant"`. If we choose to outlaw the use of the **<math>** elements in these cases, the two possible positions for the **<math>** element become indistinguishable in meaning.

2.4 Should it be possible to define a delta_variable attribute on a <role> element without also defining a stoichiometry attribute?

The combination of the **delta_variable** and **stoichiometry** attributes on a **<role>** element imply a mathematical relationship between the variable representing the general reaction rate (the *rate* variable) and the variable representing the rate of change of a particular species (the *delta* variable). There is no default value for the **stoichiometry** attribute, because the absence of this attribute will often indicate that the stoichiometry of the reaction is unknown. Currently, it is an error to define the **delta_variable** attribute without also defining the **stoichiometry** attribute (although it is legal to do the reverse, because it must be possible to define the stoichiometry for species such as activators or inhibitors, which do not have delta variables).

We could allow a **delta_variable** attribute to be defined in the absence of the **stoichiometry** attribute. There would be no mathematical relationship implied in this case. It would simply indicate to processing software which variable represents the rate of change of the species represented by the variable referenced in the containing **<variable_ref>** element. Software might choose to display the equations pertaining to this variable in a manner that indicates the special function of this variable.

The question of whether or not we should allow a **delta_variable** attribute in the absence of the **stoichiometry** attribute is related to the question of whether or not we should allow a **<math>** element in a **<variable_ref>** element or **<role>** element for which the **role** attribute value is `"product"` or `"reactant"`. Either method could be used to indicate to processing software that a certain equation or set of equations relates the general reaction rate to the rate of change of a specific species. Such a method is only necessary for cases where the math implied by the **delta_variable** and **stoichiometry** attributes would not be correct or sufficient for establishing this relationship.

2.5 Recommended resolutions for the issues raised

This section summarizes in detail the conclusions discussed in the preceding sections, and answers the issues raised in those sections. A more formal and concise discussion, which will be added to the CellML specification, is presented in Section 2.6.

- We need to allow arbitrary mathematical relationships to be defined between the rate variable and a delta variable. [see Section 2.1]

- It is useful to allow the modeller to indicate which variable is the delta variable for a given chemical species, even if he or she is using MathML to relate the delta variable to the rate variable. [see Section 2.1]
- It is useful to allow the modeller to group all of the math pertaining to a reaction inside the **<reaction>** element. [This makes the **<reaction>** element relatively reusable.]
- Storing math inside a reaction element indicates that the equations are in some way relevant to a particular variable in a particular role. Math can be stored about any variable in any role. [see Section 2.2]

Therefore we can make the following recommendations:

- The **<math>** element should be placed inside the **<role>** element, not the **<variable_ref>** element.
- The **<math>** element should be allowed in all variable references in a **<reaction>** element, regardless of the value of the **role** attribute on the **<role>** element. The equations in the **<math>** element must be relevant to the variable referenced by the containing **<variable_ref>** element in the role defined by the **role** attribute on the **<role>** element. There are three possible cases:
  - If the **role** attribute value is `"rate"`, the equations calculate the kinetic rate law (i.e., calculate the value of referenced variable).
  - If the **role** attribute value is `"reactant"` or `"product"`, the equations calculate the relationship between the general reaction rate and the rate of change of the species represented by the referenced variable (i.e., calculate the value of variable named in the **delta_variable** attribute).
  - In all other cases, the equations relate an intermediate variable used in the rate calculation to the variable referenced by the containing **<variable_ref>** element. One example of this sort of relationship is the calculation of an effective concentration of a catalyst.

  We recognize that it will be difficult to determine whether or not a CellML document adheres to these rules. We are discussing the definition of multiple levels of validity of a CellML document, and expect that any validator attempting to check these rules (perhaps by checking that the equations contain the correct variables) would only raise warnings if it thinks these rules are not obeyed. Our ideas about multiple validity levels and expected processor behavior will be discussed in more detail in Section 4.
- The **delta_variable** attribute should be required on a **<role>** element for which the **role** attribute value is either `"reactant"` or `"product"` if either of the following two conditions is met:
  - the **stoichiometry** attribute is defined (in this case, the **delta_variable** and **stoichiometry** attributes define implicit math relating the delta variable to the rate variable).
  - the **<role>** element includes a **<math>** element (in this case the equations in the **<math>** element define the relationship between the delta variable and the rate variable).

2.6 Proposed additions to the reaction portion of the specification

2.6.1 Changes to the basic structure section

The final paragraph of the basic structure section should be changed to read as follows:

The **<role>** elements may also contain **<math>** elements, which define equations using MathML. There are three uses for these equations:

- If the **role** attribute value is "rate", the equations calculate the kinetic rate law (i.e., calculate the value of referenced variable).
- If the **role** attribute value is "reactant" or "product", the equations calculate the relationship between the general reaction rate and the rate of change of the species represented by the referenced variable (i.e., calculate the value of variable named in the **delta_variable** attribute).
- In all other cases, the equations relate an intermediate variable used in the rate calculation to the variable referenced by the containing **<variable_ref>** element. One example of this sort of relationship is the calculation of an effective concentration of a catalyst.

Note that CellML processing applications are not required to be able to deduce the stoichiometry of a reaction from explicit mathematics. Therefore, it is strongly recommended that the **stoichiometry** and **delta_variable** attributes be used instead of explicit mathematics if the concentration change is simply the reaction rate multiplied by the stoichiometry. (The rules for deriving this mathematical relationship from the **stoichiometry** attribute are defined in Section ??.)

2.6.2 Changes to the document rules

The following rules should be added to or changed in the reaction portion of the specification:

Proper use of a <math> element inside a <role> element

- A **<math>** element inside a **<role>** element must define equations that are relevant to the variable referenced by the containing **<variable_ref>** element, in the role defined by the **role** attribute on the **<role>** element. [What relevant means in this context is discussed in Section ??].

In the **Proper use of the delta_variable attribute**, the second bullet point should be changed to:

- A **<role>** element on which a **delta_variable** attribute is declared must also either declare a **stoichiometry** attribute or include a **<math>** element. [The combination of the **delta_variable** attribute and the **stoichiometry** attribute implies a mathematical relationship between the variable named in the **delta_variable** attribute and the variable assigned the role of "rate", as defined in section 5.5.2. If the **stoichiometry** attribute is absent, the relationship between the variable named in the **delta_variable** attribute and the variable assigned the role of "rate" must be defined using MathML, which should be placed in the **<math>** element included in the **<role>** element.]

In the **Proper use of the stoichiometry attribute**, the second bullet point should be changed to:

- A **<role>** element on which the **stoichiometry** attribute is defined may not also include a **<math>** element. [The equations in a **<math>** element inside a **<role>** element for which the **role** attribute is "reactant" or "product" must relate the variable named in the **delta_variable** attribute to the variable assigned the role of "rate". Such equations would contradict the relationship implied by the **delta_variable** and **stoichiometry** attributes, as defined in Section 5.5.2.]

2.6.3 Changes to the behaviour rules

The following expectations for processor behavior should be added to or changed in the reaction portion of the specification:

Meaning of mathematics in a reaction element

Equations defined in a math element inside a role element have one of the following three meanings:

- If the **role** attribute value is "rate", the equations calculate the kinetic rate law (i.e., calculate the value of referenced variable).
- If the **role** attribute value is "reactant" or "product", the equations calculate the relationship between the general reaction rate and the rate of change of the species represented by the referenced variable (i.e., calculate the value of variable named in the **delta_variable** attribute).
- In all other cases, the equations relate an intermediate variable used in the rate calculation to the variable referenced by the containing **<variable_ref>** element. One example of this sort of relationship is the calculation of an effective concentration of a catalyst.

3 Further Concerns with Units

3.1 Corrections to the specification

Warren still had a few problems with the layout and conventions of the information in the units part of the CellML specification. These include:

- Should we swap the ordering of "**4.2.2 Definition of non-SI units**" and "**4.2.3 User Defined Units**" so that we have a chance to explain how the **<units>** element works before the reader is confronted with the conversions for the non-SI units.
- In the examples, we should probably only include attributes on the **<unit>** element that do not have the default value. At the minimum, we should be consistent and not, for instance, have empty **prefix** attributes, but **exponent** attributes with a value of "1".
- Some of the **<units>** elements in the two figures in the examples are duplicates. We should probably just have a single full page figure with all of these examples plus special and difficult cases like Fahrenheit/inch.
- "**4.5.1 Referencing units definitions**" doesn't belong in this part of the specification and can just be removed.
- We need to specify somewhere (probably in the behaviour section) that an empty **<units>** element has a 1-to-1 mapping with the definition of **dimensionless**.

3.2 Doh! Unforeseen problems with the exponent attribute

Currently we have specified that the value of the exponent attribute on a unit element must be an integer. This seems reasonable, but Chris Penland rightly pointed out that in data fitting, one frequently needs to be able to specify constants that have non-integer exponents for some dimension. He cited the following example. Consider the problem where he needs to fit some data relating conductance to voltage, and this is best fit by:

$$G = g_0 + g_1 V_m^{1.5}$$

If the units of g_0 are millisiemens, and the units of V_m are millivolts, then the units of g_1 are 'millisiemens / millivolts$^{1.5}$'. So let's allow the **exponent** attribute to have any real number as a value.

3.3 Specific problems with the `offset` attribute

Warren had further problems with the rules defining the **Proper use of the `offset` attribute**. Basically, they are not clear and still likely to cause confusion, unless the desired behaviour is better specified. Let's consider the use of the `offset` attribute with the aid of a practical problem. The following XML fragment contains the official definition of the non-SI unit `celsius` in terms of the SI unit `kelvin` — note that this need never be specified in a CellML document because `celsius` is in the official dictionary.

```
<units name="celsius">
  <unit units="kelvin" offset="273.15" />
</units>
```

This implies the following unit conversion equation:

$$x \text{ celsius} = (x + 273.15) \text{ kelvin}$$

Now consider the definition of the units "*metre per degrees celsius*", where we use the celsius definition from above.

```
<units name="metre_per_celsius">
  <unit units="metre" />
  <unit units="celsius" exponent="-1" />
</units>
```

This implies the following unit conversion equation, according to the spec:

$$x \text{ metre\_per\_celsius} = x \text{ (metre) (celsius)}^{-1}$$

Before we even consider how that might then be further simplified to the SI units, consider the same definition where we define *per degrees celsius* inside the units definition.

```
<units name="metre_per_celsius">
  <unit units="metre" />
  <unit units="kelvin" exponent="-1" offset="273.15" />
</units>
```

We currently have no facility for dealing with the `offset` attribute on the second `<unit>` element, yet we have declared that it is legal in both the specification and the Royal Society Paper.

Lets think about this a bit more. In units definitions that involve an offset, we are typically thinking of a ratio variable. In the case of `metre_per_celsius`, we might be saying that a substance expands by x metres for every degree celsius that its temperature is raised. There are no formulae that state that a substance has a size of x metres at y degrees celsius, because celsius has an arbitrary zero level (hence the offset).

What we really want to do is specify that an application is free to ignore the `offset` attribute on a `<unit>` element inside a complex unit definition. This simplifies the rules considerably — in fact, we can just delete the "**Proper use of the `offset` attribute**" section — and allows the user to combine pre-defined units without worrying about offsets.

We then need to copy the equations expressing units definitions into a subsection within the "**Rules for processor behaviour**" section that looks something like the following, where the new stuff is in italics.

3.3.1 The meaning of units definitions

If there is a single **<unit>** element within a **<units>** element, that **<units>** element then defines a straightforward conversion according to the formula below. *This is known as a* 'simple units' *definition.* 'Units' refers to the units being defined *and the other terms refer to the values of the attributes with the same names on the* ***<unit>*** *element.*

$$x \text{ Units} = (\text{multiplier } x + \text{offset})(\text{prefix units})^{\text{exponent}}$$

'*Complex units*' *[delete this: such as millimoles/litre]* are the product of multiple base quantities, and are created by placing several **<unit>** elements inside a single **<units>** element. The conversion between the new units and the constituent base units is given in the formula below. *As above, 'Units' refers to the units being defined.* m1, p1, u1 *and* e1 *refer to the values of the* ***multiplier***, ***prefix***, ***units*** *and* ***exponent*** *attributes on the first* ***<unit>*** *element respectively. Similarly* m2 *refers to the value of the* ***multiplier*** *attribute on the second unit element and so on. Note that the values of any* ***offset*** *attributes that appear on the* ***<unit>*** *elements in complex units definitions are ignored.*

$$x \text{ Units} = x\ [\text{m1 (p1 u1)}^{e1}]\ [\text{m2 (p2 u2)}^{e2}]\ \ldots\ \ [\text{mn (pn un)}^{en}]$$

3.4 *Special* Units

Note that this section has changed substantially between the current and original versions, as a lot of issues were not seen at the time of writing of the original version. These issues resulted in some substantial additions to this section.

Sometimes we might want to define and use units for which no simple conversion to SI units exist. When using such units we want to indicate that they may not be mapped to any other type of unit, other than one with the same name. A good example of this is pH, which is dimensionless, but uses a log scale. We can't simply define pH as dimensionless because software might then think that this could then be simply converted into any other dimensionless quantity.

One might argue that we can't prevent the modeller from being a complete idiot all the time, but this functionality allows smart modellers to prevent dumb modellers from doing silly things with their models. One might also claim that it doesn't make models and components re-usable because a variable that references units defined with the map-by-name-only flag can only be mapped to variables that reference the same units definition, or another units definition that looks identical. Again, it is useful in preventing mistakes, because it allows modellers to flag that the units for a certain variable have special properties, and it needs to be given special attention. The specification and any tutorial documentation should recommend against the widespread use of this feature.

The most simple implementation of this feature is to introduce an attribute on the units element that is associated with the required behaviour. The name of this attribute could be any of **mappable** or **map_by_name_only** or **map_by_name** — we'll go with the last one for the right mix of conciseness and clarity. (Melanie also suggested the somewhat ridiculous **name_mapping**.) Its value could be "yes" or "no" and would default to "no", implying that variables using units declared without the attribute could be mapped to any other variable with units of the same dimensionality. Its use is shown below.

```
<units name="pH" map_by_name="yes" />
```

The introduction of the **map_by_name** attribute introduces some other non-obvious problems. For instance, can these units be used in the definition of complex units, such as pH/metre. The easiest answer is just to forbid it outright, forcing the modeller to define new non-mappable units for each new quantity. Problem solved! Another problem is that we have to consider if it is useful to allow **<unit>** elements inside a **<units>** element with a **map_by_name** attribute value of "yes". This probably isn't useful

so we can just make it illegal. Another problem is that it becomes difficult to use variables that use non-mappable units inside equations. Suppose we want to convert a variable representing the concentration of hydronium ions into another variable defined using units of **pH**. We need to insert unitary numbers with the appropriate units to balance out the equations, as shown in Figure 2.

```
<component name="aqueous_solution">

  <units name="concentration">
    <unit prefix="milli" units="mole" />
    <unit units="litre" exponent="-1" />
  </units>

  <units name="pH" map_by_name="yes" />

  <variable name="H3O" units="concentration" />
  <variable name="pH" units="pH" />

  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply><eq />
      <ci> pH </ci>
      <apply><times />
        <cn cellml:units="pH"> 1.0 </cn>
        <apply><log />
          <apply><divide />
            <ci> H3O </ci>
            <cn cellml:units="concentration"> 1.0 </cn>
          </apply>
        </apply>
      </apply>
    </apply>
  </math>

</component>
```

FIGURE 2: When variables defined in terms of non-mappable units are used in an equation, we need to introduce unitary numbers to balance the equation properly.

The addition of the **map_by_name** attribute affects several parts of the specification. First of all, the contents of the units element are now constrained by its attributes, so under **Allowed use of the <units> element**, we need:

- If a **<units>** element defines a **map_by_name** attribute with a value of "yes", then that **<units>** element must be empty.
- If a **<units>** element does not define a **map_by_name** attribute with a value of "yes", then that **<units>** element may contain any number of **<unit>** elements.

We then need a new section called **Allowed values of the map_by_name attribute** with the following rules:

- If present, the value of the **map_by_name** attribute must be "yes" or "no".
- If not present, the value of the **map_by_name** attribute defaults to "no".

I think we need to add a subsection under the "**Rules for processor behaviour**" section called "**Wiring components together**". This might look like the section below. It might also be worth including a section on equation consistency checking, although I'm not sure if we actually expect software to do that. Note that these kinds of sections would come under level two conformance as discussed in Section 4.

3.4.1 Wiring components together

One of the goals of providing facilities for the definition of complex and arbitrary sets of units is to enable the combination of components and models defined using different units sets into a single model. This can be made to work if all units are defined in terms of a single dictionary. CellML processing software is expected to be able to break down the definitions of simple units that involve an offset into their dimensionality in terms of the SI units plus some conversion operations, and complex units definitions into their dimensionality plus some conversion factor. Any two variables that share the same dimensionality may be connected, and CellML processing software is expected to allow this, and perform the appropriate conversions when running simulations or generating code.

The exception to the rule occurs when the `map_by_name` attribute on the `<units>` element representing a units definition has a value of `"yes"`. In this case, software may not allow variables that reference that units definition to be mapped to any variable that does not reference that units definition or one with exactly the same name.

4 Types Of Rules

The CellML specification currently consists of lots of little rules. Some of these rules define the way in which elements and attributes can be combined in a CellML document. Others define some of the behaviour that we expect processing software to imply from a CellML document — an example is the implied math generated by the use of the `delta_variable` and `stoichiometry` attributes on a `<role>` element. Currently we have split these rules into two categories: 'Rules For Use' and 'Rules For Processor Behaviour' (where the first one probably needs to be re-named at some point to, for instance 'Rules For Documents').

It becomes apparent when we consider more complicated rules like 'all equations in a model must be consistent', or 'the units on variables and numbers in an equation must balance out', that we are expecting quite a lot from CellML processing software. It has been proposed that we create several layers of validation for CellML documents, and formalize this in the specification. Section 4.1 is a candidate for inclusion in the specification. Current rules that might be in level two as described in Section 4.1 are:

- those that relate to the breakdown and checking of units for variables that are mapped together
- those that relate to the breakdown and checking of units within equations
- those that relate to the consistency checking of equations throughout a model

4.1 Levels of CellML Conformance

The rules in the CellML specification can be split into two groups: those that limit the format of a CellML document, and those that determine how software processing that document should behave. In the subsequent sections of the specification, the first set of rules are included in subsections titled 'Rules For Documents', and the second in subsections titled 'Rules for Processor Behaviour'.

The rules can also be split into two groups, each representing different levels of conformance to the specification. The majority of the rules in the CellML specification are part of the first level of conformance. Where rules are part of the second level of conformance, this is highlighted. What these conformance levels mean for documents and processing software is discussed below:

4.1.1 CellML Conformance Level One

The first level of CellML conformance is composed of the majority of rules in the CellML specification. Level one document rules generally specify how the different XML elements and attributes that make up the CellML vocabulary may be combined. A typical level one rule for a document is 'Both the **<model>** and **<component>** elements can contain any number of **<units>** elements'. Level one processor rules generally specify implied behaviour that is not immediately obvious from analysis of the XML. A typical level one rule for processor behaviour defines the scope of a units definition (see Section ??).

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 worry about the scope of units definitions).

4.1.2 CellML Conformance Level Two

The second level of CellML conformance rule is composed of all of the rules from level one plus additional rules that are marked as belonging to level two in the specification. Currently, there are no level two document rules. Level two processor rules generally specify complex interactions between objects defined in different parts of a CellML document. A typical example is the requirement that all mathematics within a model be self-consistent.

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 worry about the consistency of mathematics).