Meeting Minutes 16 July 2001

Equation Scope and Initial Conditions

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Autumn Cuellar (Bioengineering Institute, University of Auckland)
Poul Nielsen (Bioengineering Institute, University of Auckland)

1 Introduction

This document resulted from a meeting on July 16, was used as input to a meeting on July 17, and then was substantially revised based on that discussion.

This document deals with two under-defined areas in the CellML specification: equation scope and initial conditions. Equation scope is the domain in which an equation can be said to "hold" or be relevant. This is not normally specified for the equations in a CellML model, which typically are assumed to hold at all times and locations in a simulation, but becomes a problem when combined with explicitly specified initial conditions. Initial conditions themselves are not very elegantly handled in CellML, and in particular the definition of the **`initial_value`** attribute on the **`<variable>`** element needs a bit more clarification.

2 Equation Scope

In the current set of examples available on the CellML website, it is never specified what the scope of particular equations is. It is normally just assumed that the domain of solution for a model that has time as the independent variable is $t > 0$. In part, this represents a conscious decision on the part of CellML's authors that simulation information should be kept separate from the model itself (a very fuzzy boundary), and in part under-specification of the mathematics. In fact, a model's definition is very closely coupled with the kind of simulations that may be run, but the fact that many different simulations may need to be run with a given model demands that this information be separated.

It is not currently specified, but should be, that the scope of an equation in a CellML document is infinite with respect to all independent variables, unless specifically constrained using the appropriate MathML elements. If all equations have infinite scope, then it becomes simple to define invalid math with contradictory equations.

A case that highlights the problem is the setting of an initial condition for a variable based on evaluation of other parameters, as shown in the trivial example in Figure 1, where the first two equations are initial conditions only. When combined with a differential equation, with neither equation containing any sort of conditional status, the result is two equations with overlapping scope.

The MathML in Figure 1 defines the following three equations:

$$constant\_a = 5.0 \tag{1}$$

$$dependent\_var = constant\_a \cdot 2.0 \tag{2}$$

$$\frac{d\ dependent\_var}{d\ time} = \frac{constant\_a}{2.0} \tag{3}$$

```
<component
    name="dependent_initial_condition_example"
    xmlns="http://www.cellml.org/cellml/1.0#"
    xmlns:cellml="http://www.cellml.org/cellml/1.0#">
  <variable name="constant_a" units="metre" />
  <variable name="dependent_var" units="metre" />
  <variable name="time" public_interface="in" units="second" />

  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply><eq />
      <ci> constant_a </ci>
      <cn cellml:units="dimensionless"> 5.0 </cn>
    </apply>

    <apply><eq />
      <ci> dependent_var </ci>
      <apply><times />
        <ci> constant_a </ci>
        <cn cellml:units="dimensionless"> 2.0 </cn>
      </apply>
    </apply>

    <apply><eq />
      <apply><diff />
        <ci> dependent_var </ci>
        <bvar><ci> time </ci></bvar>
      </apply>
      <apply><divide />
        <ci> constant_a </ci>
        <cn cellml:units="second"> 2.0 </cn>
      </apply>
    </apply>
  </math>
</component>
```

FIGURE 1: An example containing three equations with overlapping scope. Note that, although this is valid CellML, **the mathematics are invalid**.

In this particular problem, we will assume that time is the only independent variable. If all three equations are assumed to have infinite scope and **constant_a** is non-zero (which it is), then the three equations can not be simultaneously satisfied for all values of time.

As previously stated, the second equation is an initial condition, which holds for **time** = 0.0 only. Let's examine how we can legally specify this in MathML.

MathML 2.0 provides several constructs with which the scope of an equation can be limited. Only one of these is included in the CellML subset of MathML elements defined in Section 4.2.3 of the 18 May 2001 spec[1]. That is the **<piecewise>**, **<piece>**, **<otherwise>** construct, which allows the definition of switches and basic conditional statements. It would be tempting to try and describe the problem with the MathML shown in Figure 2, which limits the scope of each equation.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <piecewise>
    <piece>
      <apply><eq />
        <ci> dependent_var </ci>
        <apply><times />
          <ci> constant_a </ci>
          <ci cellml:units="dimensionless"> 2.0 </ci>
        </apply>
      </apply>
      <apply><leq />
        <ci> time </ci>
        <cn cellml:units="second"> 0.0 </cn>
      </apply>
    </piece>
    <otherwise>
      <apply><eq />
        <apply><diff />
          <ci> dependent_var </ci>
          <bvar><ci> time </ci></bvar>
        </apply>
        <apply><divide />
          <ci> constant_a </ci>
          <cn cellml:units="second"> 2.0 </cn>
        </apply>
      </apply>
    </otherwise>
  </piecewise>
</math>
```

FIGURE 2: The mathematics from Figure 1 might be rewritten using MathML's **<piecewise>**, **<piece>**, **<otherwise>** construct to prevent the scope of the two equations from overlapping, and hence the equations from contradicting each other. **We believe that this is an invalid use of MathML**.

The MathML in Figure 2 defines the following case statement, where the case statement has no LHS:

$$\begin{cases} dependent\_var = constant\_a \cdot 2.0 & \text{if } time = 0 \\ \frac{d\ dependent\_var}{d\ time} = \frac{constant\_a}{2.0} & \text{otherwise} \end{cases} \quad (4)$$

[1] http://www.cellml.org/public/specifi cation/20010518/mathematics.html#sec_math_cellml_subset

It states in Section 4.4.2.16 of the MathML 2.0 specification[2] that the **<piecewise>** element can only be used as a constructor. This suggests (although it is not explicitly stated) that the **<piecewise>** element constructs some value, which can then be assigned to something. We believe that the use of the **<piecewise>** element in Figure 2 is invalid, or at least does not conform to the intent of the authors of the MathML specification.

A problem like this is better handled using the **<forall>** element, which is discussed in Section 4.4.3.17 of the MathML 2.0 specification[3]. The **<forall>** element allows the modeller to conveniently specify the scope of an assertion in terms of bound variables and a condition statement. Unfortunately for us there is no way of connecting two "forall" statements to indicate that they are mutually exclusive, as a "piecewise" statement does. So the correct definition of the dependent initial condition problem is shown in Figure 3.

The equations defined in Figure 3 are shown below. The specification of equation scopes allows a solver to unambiguously determine which equation to evaluate for any given value of the independent variable **time**, removing any potential conflicts from the model definition.

$$constant\_a = 5.0 \tag{5}$$

$$dependent\_var = constant\_a \cdot 2.0 \qquad \forall\ time = 0 \tag{6}$$

$$\frac{d\ dependent\_var}{d\ time} = constant\_b \qquad \forall\ time > 0 \tag{7}$$

Note that the **<forall>** element is not in the CellML subset of MathML elements. This was deliberately omitted because the authors don't anticipate much use for this element, and its use involves a reasonably significant burden for the implementors of CellML processing software.

3 Initial Conditions

In the trivial example shown in Figure 1, it was apparent that the initial condition for the **dependent_var** could actually be set using an **initial_value** attribute on the **<variable>** element. The meaning of this attribute is currently poorly defined in the CellML specification with the following text:

> **initial_value** — This attribute provides a convenient means for specifying the initial or default value of a scalar variable in a simulation with time as the independent variable. The variable's value may be reset or modified in equations in the current component. The initial values of variables need not be set in the model definition at all; they could instead be set in a configuration file loaded separately by the model processor.

The problem of specifying initial and/or boundary conditions is so huge that it was deliberately left out of CellML 1.0 to make CellML manageable. In any simulation with more than one independent variable, something better than a set of scalar values will generally be need to specify a model's boundary conditions. This could use MathML to specify equations along a boundary, or FieldML to specify a piecewise polynomial approximation. In problems with more than one independent variable, the initial and/or boundary conditions will invariably be closely linked to the simulation itself, which seems like a good argument for leaving this problem to be dealt with when we finally come to standardising a simulation specification format.

[2] http://www.w3.org/TR/MathML2/chapter4.html#contm_piecewise
[3] http://www.w3.org/TR/MathML2/chapter4.html#contm_forall

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply><eq />
    <ci> constant_a </ci>
    <cn cellml:units="dimensionless"> 5.0 </cn>
  </apply>

  <apply><forall />
    <bvar><ci> time </ci></bvar>
    <condition>
      <apply><eq />
        <ci> time </ci>
        <cn cellml:units="second"> 0.0 </cn>
      </apply>
    </condition>
    <apply><eq />
      <ci> dependent_var </ci>
      <apply><times />
        <ci> constant_a </ci>
        <cn cellml:units="dimensionless"> 2.0 </cn>
      </apply>
    </apply>
  </apply>

  <apply><forall />
    <bvar><ci> time </ci></bvar>
    <condition>
      <apply><gt />
        <ci> time </ci>
        <cn cellml:units="second"> 0.0 </cn>
      </apply>
    </condition>
    <apply><eq />
      <apply><diff />
        <ci> dependent_var </ci>
        <bvar><ci> time </ci></bvar>
      </apply>
      <apply><divide />
        <ci> constant_a </ci>
        <cn cellml:units="second"> 2.0 </cn>
      </apply>
    </apply>
  </apply>
</math>
```

FIGURE 3: In this MathML fragment, the **<forall>** element is used to limit the scope of two of the equations, preventing the second two equations from conflicting.

However, for many of the simple models that we deal with now, with only a single independent variable time and only ordinary differential equations, it is sufficient to specify an initial condition and a differential equation for each state variable, and run meaningful simulations from this. The **initial_value** attribute is a convenient way for embedding the initial conditions on any variable for such a simulation in the model definition. It also allows the specification of constant variables — it is often appropriate to store constants in variables so that they can be defined in one place only and then referenced by name, and also to indicate to software that the constant can be modified by the user.

The CellML specification needs a better definition of the **initial_value** attribute, and CellML processing software needs some guidelines on how to handle it in various cases. One possible option was to define the **initial_value** attribute's semantics and the associated processor behaviour as follows:

- The attribute provides a convenient means for specifying the initial or default value of a scalar variable in a simulation with a single independent variable (as opposed to time specifically — CellML processing software may not assume that a particular variable represents time, even if it is called **time**).
- CellML processing software must ignore the value of an **initial_value** attribute on a **<variable>** element in a model that has more than one independent variable. (The alternative is to say that it is an error to define an **initial_value** attribute in such models — any ideas?)

This particular solution makes components less re-usable, as when a component where variables are declared with **initial_value** attributes are moved from a model with one independent variable to a more complex model with more than one independent variable, the initial conditions would be lost. A better solution is to say that:

- The **initial_value** attribute provides a convenient means for specifying the value of a scalar variable when all independent variables in the model have a value of 0.0.
- CellML processing software must assume that the presence of an **initial_value** attribute on a variable element implies the following equation:

$$variable = initial\_value \qquad \forall \ independent\ variables = 0.0 \tag{8}$$

4 Changes to the Spec

The points raised above merit some changes to the CellML specification in the interests of clarification of these potentially confusing issues. I propose the following changes to the unstable version of the CellML specification available on 16 July 2001 (see the corrections to the 18 May 2001 final draft[4] before this date to get an idea of what it may have looked like, if you come across this document later).

After "Section 4.5.1 - Ordering of Expressions", we should add "**Section 4.5.2 - Scope of Expressions**", with the following text:

> CellML processing software must make no assumptions about the scope or domain of a mathematical expression defined within a model. Unless explicitly stated, all expressions hold for any and all combinations of independent variables.

After "Section 4.2.4 - Ordering of Expressions", we should add "**Section 4.2.5 - Scope of Expressions**", with the following text:

[4] http://www.cellml.org/public/specifi cation/20010518/errata.html

> Within a CellML model, all expressions are assumed to have unlimited scope with respect to the independent variables unless explicitly stated using MathML's **<piecewise>** construct or some other form of conditional expression. This means that if the initial conditions for a variable, the value of which is determined by a differential equation, are to be specified using an equality, the two equations should have their scope limited so that they do not contradict each other.

In "**Section 3.2.3 - Definition of variables**", the bullet point describing the **initial_value** attribute should be changed to have the following text:

- The **initial_value** attribute provides a convenient means for specifying the value of a scalar variable when all independent variables in the model have a value of 0.0. Independent variables are those with respect to which another variable is differentiated or integrated.

Directly under the bulleted list, the following explanatory text should be added:

> The name of the **initial_value** attribute derives from the fact that, in a model with only one independent variable, this would generally correspond to time, and so the value of the **initial_value** attribute set the starting condition for a simulation which progressed from time equals 0.0. The initial values of variables need not be set in the model definition at all. When multiple simulations are to be run using the same model, initial and boundary conditions are most conveniently set in an external simulation configuration file loaded separately by CellML processing software.

I don't think we actually need to specify any processing software rules relating to the **initial_value** attribute.
